

ME850 单片机开发实验仪

使 用 手 册

深圳硕飞科技有限公司

SOFI TECHNOLOGY CO., LTD.

TEL: +86-755-8486 7757
FAX: +86-755-8486 7941
WEB: WWW.SOFI-TECH.COM
Email: SOFITECH@TOM.COM

技术支持：WWW.MCUSJ.COM（伟纳单片机世界）

Publication Release Date: October 2008
Revision A1

目　　录

第一章 产品介绍



- 国内首款集成 USB2.0 专业仿真器/编程器(ICE52)
- 增强型仿真功能，零资源占用，无限制真实仿真
- 集成 ISP 编程功能，支持对外部 ISP 下载编程
- 专业编程控制软件，国内首创烧写功能直接嵌入 Keil
- 全开放模块化设计，可快速跳线连接和杜邦线自由搭配
- 可以支持编程和实验开发的芯片全（51/AVR 两大系列）
- 丰富汇编与 C 语言实验例程及详细的全程实验指导
- USB 与外接双重电源选择
- 完善的过流/短路保护功能
- 高档铝合金外箱，使用携式方便

1.1 产品简介

ME850 单片机开发实验仪是深圳硕飞科技有限公司集多年单片机开发工具设计经验，最新研发的具有"实验仪、编程器、仿真器、ISP 下载线"四功能合一的新一代单片机开发实验系统。ME850 系列单片机开发实验仪集成了真正 USB2.0 接口的专业仿真/编程器 ICE52，仿真器不占用用户资源，无限制真实仿真（32 个 IO、串口、T2 可完全单步仿真），支持全系列标准 8051 芯片仿真。系统支持目前大多数主流公司（如 ATMEL，Winbond，NXP（Philips），SST，DALLAS）的 51 系列单片机的烧写和实验，同时也兼容 AVR 系列单片机的烧写和实验。

ME850 板载丰富的实验硬件资源和接口，各个功能模块各自独立，并对外全部开放 I/O 口，既可简单地使用短路帽进行默认的资源连接（方便初学者直接使用），也可以取下短路帽后采用杜邦线连接单片机的任意 I/O 口线，轻松搭建自己的电路。配合本公司精心编写的大量 C 语言与汇编实验例程，可使用户快速掌握单片机原理及其实用接口技术。

ME850 单片机开发实验仪适合单片机初学者，院校学生学习单片机使用。专业的仿真功能和编程下载功能也真正适合单片机工程师开发单片机产品使用。



ME850 单片机开发实验仪主机

1.2 硬件布局

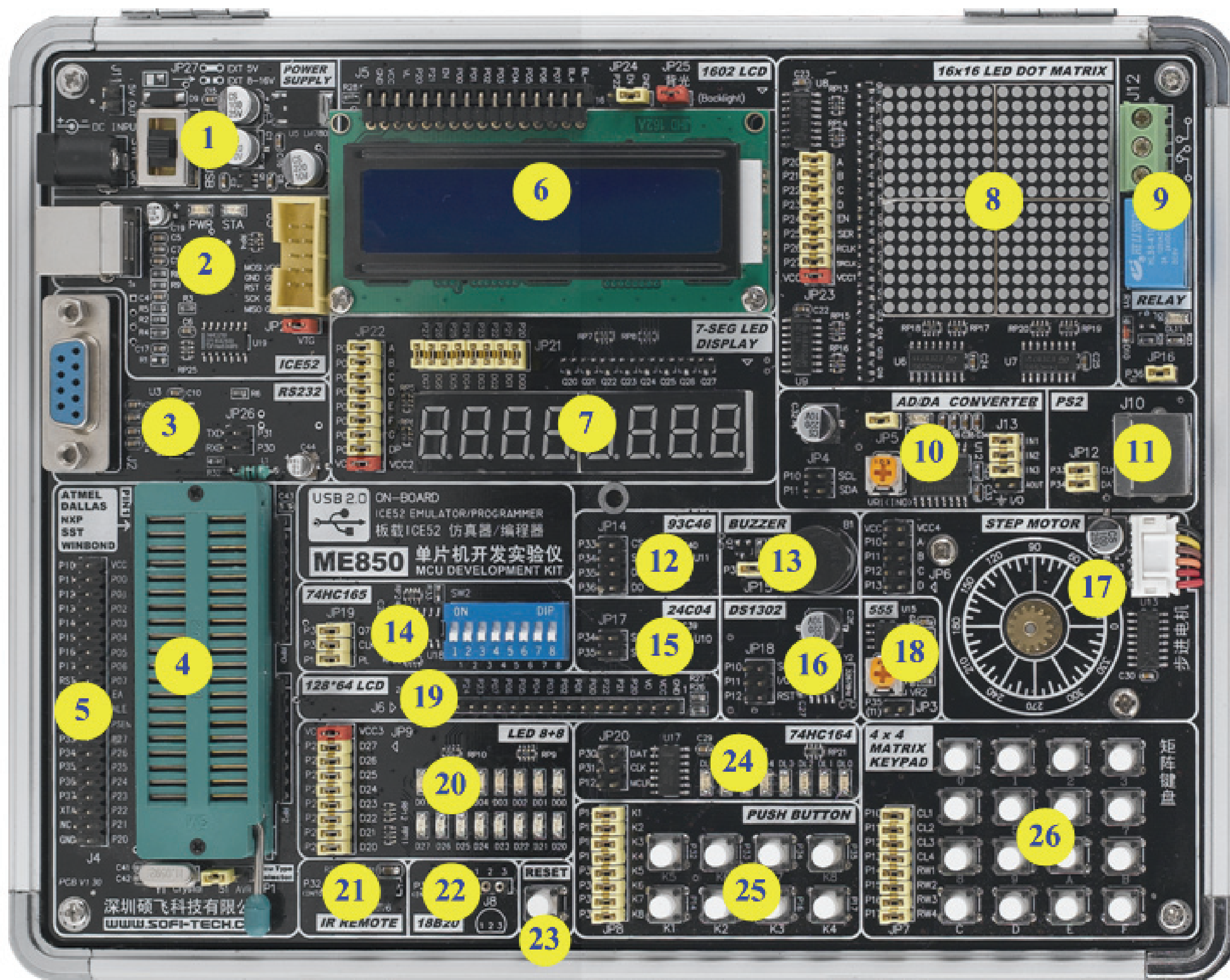
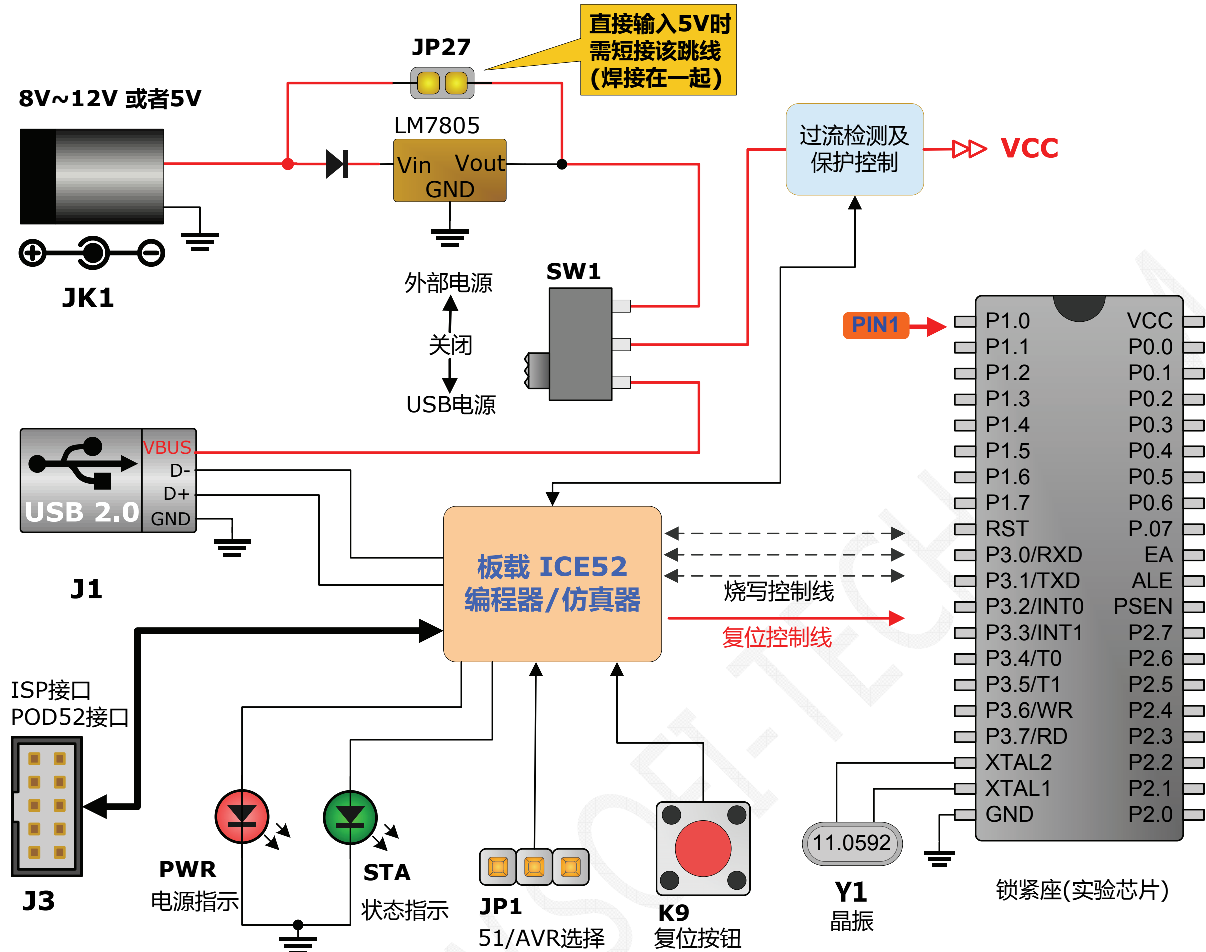


图 4.1

编号	说　　明	编号	说　　明
1	电源部分	14	74HC165 并入串出端口扩展
2	增强型 ICE52 仿真器/编程器	15	24C04 存储器
3	DB9 串行接口及 MAX232 电平转换器	16	DS1302 实时时钟 RTC
4	锁紧座(用于放置实验芯片或仿真适配头)	17	步进电机
5	40PIN 扩展端口	18	NE555 时脉发生器
6	字符型液晶显示模式 LCD1602	19	128x64 图像液晶模块接口
7	8 位 7 段数码显示器	20	16 路发光二极管
8	16x16 点阵显示屏	21	一体化红外线接收器
9	1 路继电器	22	DS18B20 一线温度传感器接口
10	模数/数模转换器	23	系统复位按钮
11	PS2 键盘接口	24	74HC164 串入并出端口扩展
12	93C46 存储器	25	8 路建议按钮
13	蜂鸣器	26	4x4 矩阵键盘

1.3 电源及系统控制



- ME850 支持两种供电方式，分别为 USB 接口供电和外部供电．板上电源开关 SW1 可用于选择供电方式，将开关拨到 USB 方，选择 ICE52 的 USB 接口供电．将开关拨到 EXT．方向选择外部电源，即由 JK1 接口电源．
- 在使用外部供电时,开发板不仅可以使用 8V~12V 直流进行供电，还可以直接使用 5V 进行供电．ME850 板上已经带有 LM7805 电源稳压器，可接受 8V~12V 直流输入电压．电源接入到 JK1 的极性必须是内正外负．
- 使用外部直接 5V 稳压供电时，需将开发板上 JP27 的两个焊点焊接在一次．默认焊点处于开路状态
- ME850 内置有电源过流保护系统，如果用户不慎放反实验芯片或接错资源时造成电源过载时，控制芯片会立即切断实验部分的供电系统，有效地保证开发板以及实验芯片不受损坏．在过流保护状态，ICE52 的电压指示灯以及状态指示灯将不停的闪烁．以警示用户目前开发板有过流情况．在用户排除故障后，重新开启电源(SW1 先置于关闭位置，再拨到相应的电源)，系统部分才会重新向实验部分供电．
- ME850 可兼容 51 与 AVR，在使用之前需通过 JP1 设置合适的单片机类型．
- 复位按钮 K9 用于复位实验芯片．
- J3 是 ISP 接口/POD52 接口，它有两个作用，一是用于连接 POD52 仿真适配头，二是用户对外部的芯片进行 ISP 编程操作．

特别注意： 使用 8-16V 的直流电源：必须确保 JP27 焊盘断开，否则将会烧坏 ME850；
使用 5V 的稳压电源：　　需要短接 JP27 焊盘，否则将导致供电不足；
ME850 可以直接使用 USB 接口供电，可以不使用外接电源供电。

1.4 资源端口分配

端口	资源(1)	资源(2)	资源(3)	资源(4)	资源(5)
P0.0	流水灯(D00)	数码管(段 a)	LCD1602(D0)	LCD12864(D0)	
P0.1	流水灯(D01)	数码管(段 b)	LCD1602(D1)	LCD12864(D1)	
P0.2	流水灯(D02)	数码管(段 c)	LCD1602(D2)	LCD12864(D2)	
P0.3	流水灯(D03)	数码管(段 d)	LCD1602(D3)	LCD12864(D3)	
P0.4	流水灯(D04)	数码管(段 e)	LCD1602(D4)	LCD12864(D4)	
P0.5	流水灯(D05)	数码管(段 f)	LCD1602(D5)	LCD12864(D5)	
P0.6	流水灯(D06)	数码管(段 g)	LCD1602(D6)	LCD12864(D6)	
P0.7	流水灯(D07)	数码管(段 dp)	LCD1602(D7)	LCD12864(D7)	
P1.0	矩阵键盘(列 1)	步进电机	AD/DA(SCL)	DS1302(SCLK)	
P1.1	矩阵键盘(列 2)	步进电机	AD/DA(SDA)	DS1302(I/O)	
P1.2	矩阵键盘(列 3)	步进电机	HC164(MCLR)	DS1302(RST)	
P1.3	矩阵键盘(列 4)	步进电机		HC165(PL)	
P1.4	矩阵键盘(行 1)	按键(K1)			
P1.5	矩阵键盘(行 2)	按键(K2)			
P1.6	矩阵键盘(行 3)	按键(K3)			
P1.7	矩阵键盘(行 4)	按键(K4)			
P2.0	流水灯(D20)	数码管(位 0)	LCD1602(RS)	LCD12864(D/I)	16x16 点阵(行选)
P2.1	流水灯(D21)	数码管(位 1)	LCD1602(R/W)	LCD12864(R/W)	16x16 点阵(行选)
P2.2	流水灯(D22)	数码管(位 2)	LCD1602(EN)	LCD12864(E)	16x16 点阵(行选)
P2.3	流水灯(D23)	数码管(位 3)		LCD12864(CS1)	16x16 点阵(行选)
P2.4	流水灯(D24)	数码管(位 4)		LCD12864(CS2)	16x16 点阵(列)
P2.5	流水灯(D25)	数码管(位 5)		LCD12864(RSTB)	16x16 点阵(列)
P2.6	流水灯(D26)	数码管(位 6)			16x16 点阵(列)
P2.7	流水灯(D27)	数码管(位 7)			16x16 点阵(列)
P3.0(RXD)	串口(RXD)		HC164(DAT)	HC165(DAT)	
P3.1(TXD)	串口(TXD)		HC164(SCK)	HC165(SCK)	
P3.2(INT0)	红外线接收	按键(K5)			
P3.3(INT1)	DS18B20(DQ)	按键(K6)	93C46(CS)	PS2(CLK)	
P3.4(T0)	24C04(SCL)	按键(K7)	93C46(SK)	PS2(DATA)	
P3.5(T1)	24C04(SDA)	按键(K8)	93C46(DI)	NE555(Q)	
P3.6(WR)	继电器		93C46(DO)		
P3.7(RD)	蜂鸣器				

第二章 软件安装

2.1 安装 Keil C51 软件

Keil C51 是德国知名软件公司 Keil（现已并入 ARM 公司）开发的基于 8051 内核的微控制器软件开发平台，是目前开发 8051 内核单片机的主流工具。

1. 将产品附带的光盘放入光驱，运行光盘目录Software下的c51v812.exe，软件出现如下图所示的对话框

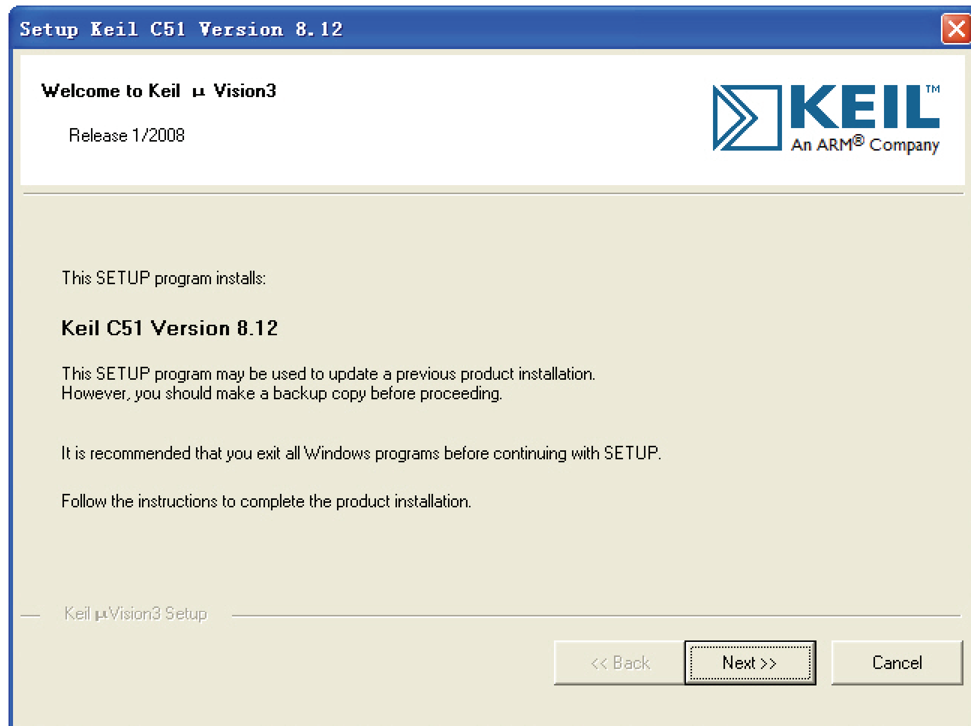


图 2.1 Keil 安装

2. 在接下来的几个对话框中，点击 Next 按钮，在提示输入用户名称和公司名时，按需要填写即可
3. 安装完成后，按 Finish 结束

2.2 ICE52 仿真驱动程序

运行产品光盘 Software 目录下的 ICE52_DLL_SETUP. Exe 软件。**注意，在安装仿真驱动程序之前必须保证电脑上已经安装有 Keil 相应版本。**安装步骤如下：

1. 启动 ICE52_DLL_SETUP 安装程序，如下图所示



图 2.2 安装 Keil 接口驱动

2. 在安装的的第 3 个画面，提示选择语言。可选择简体中文或英文。此选项决定在 Keil 中的仿真/编程对话框使用哪个语言，请根据需要进行选择。如图 2.3 所示：

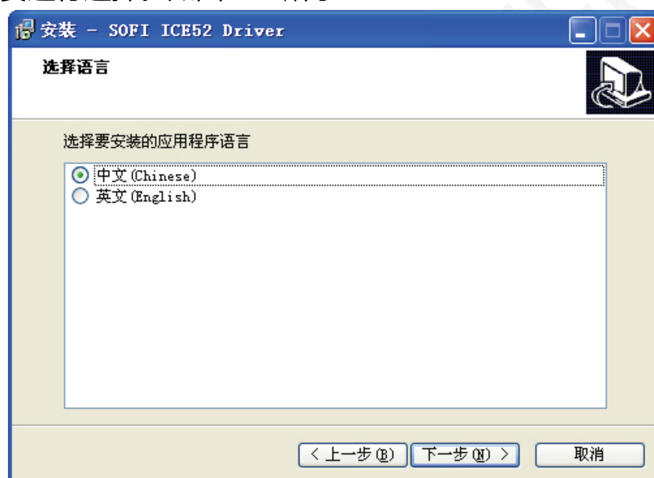


图 2.3 选择 ICE52 使用的语言

3. 然后选择 Keil 的安装目录，目录必须正确。如果在选定的目录没有检测到 Keil 软件，软件会给出如图 2.5 所示的提示。



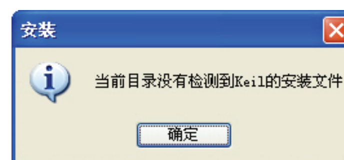


图 2.4 选择 Keil 目录

图 2.5 没有检测到 Keil

4. 选择正确的 Keil 安装目录之后，便可开始接口程序的安装。



图 2.6 安装

2.3 安装 MEFlash 软件

MEFlash 是针对 ME800 系列的单片机开发实验仪而特别开发的一款增强型编程软件. 通常在开发阶段可以直接使用 ICE52 所独特的 Keil 中集成下载功能, 而无需使用该软件. 通常在需要单独烧写代码文件, 或者是对 AVR 系列单片机进行编程才需要使用该软件.

安装程序位于产品光盘的 Software 目录下, 文件名为 MEFlash_Setup.exe. 运行该程序将出现如下画面. 直接点击"下一步", 根据安装软件的提示安装即可.



图 2.7 安装 MEFlash

2.4 安装 USB 驱动

ICE52 采用 USB 接口进行通讯，在使用之前必须安装 USB 驱动程序。安装步骤如下.

1. 用 USB 电缆连接 ME800 系列开发试验仪到电脑主机的 USB 接口, 并开启 ME800 上的电源开关(将 SW1 拨到 USB 端). 电脑会自动检测到新的硬件设备（参见如图 2.8）. 在弹出的对话框（参见图 2.9）中选择"否, 暂时不(T)", 然后点击下一步。



图 2.8

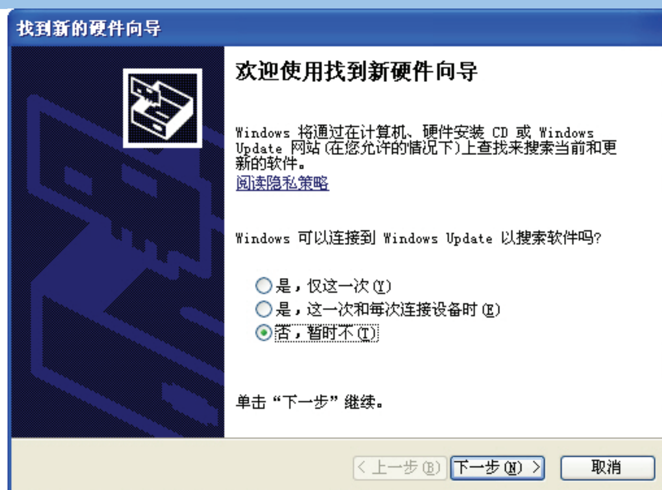


图 2.9

2. 接下来选择"从列表或指定位置安装(高级)(S)" 然后点击下一步. 参见图 2.10.

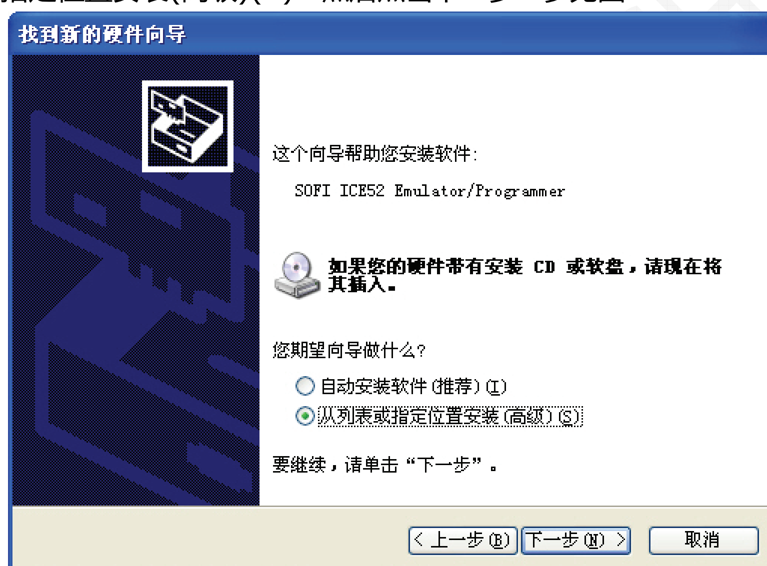


图 2.10

3. 在搜索位置内输入产品光盘目录，USB 驱动程序位于光盘中的 Software\USB_Driver 目录下. 然后点击下一步，参见图 2.11。

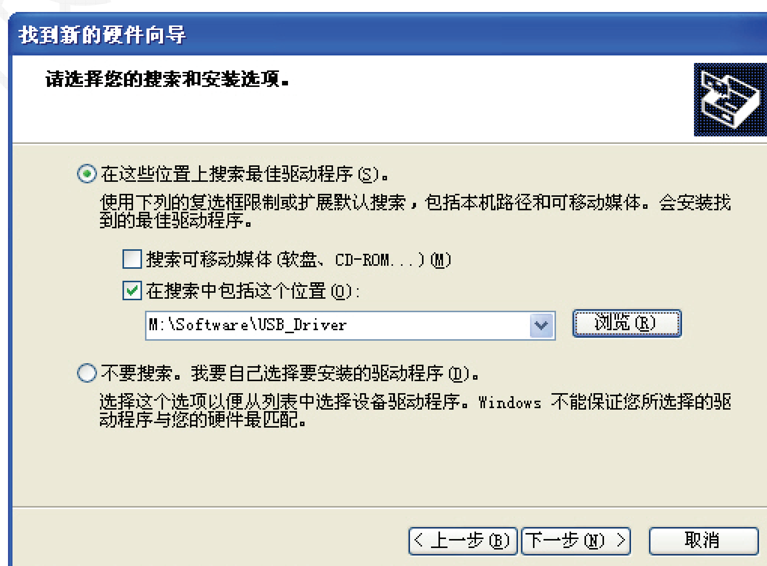


图 2.11

4. 系统开始处理驱动程序的安装，参见图 2.12。



图 2.12

5. 安装过程中可能会提示驱动程序没有通过 Windows 的微标测试，此时请选择"仍然继续"，参见图 2.13。

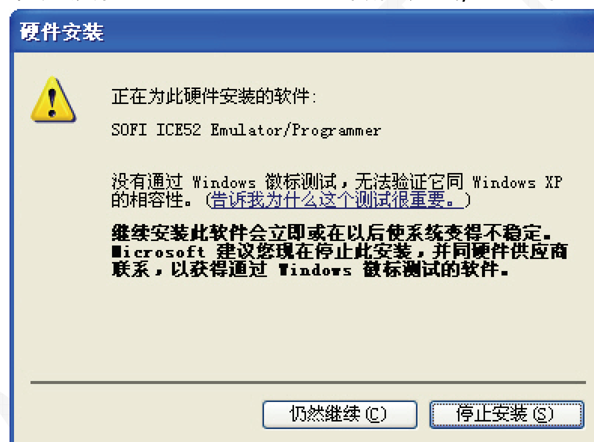


图 2.13

6. 安装完成



图 2.14

第三章 快速入门

本章将以一个 LED 闪烁程序讲述用 Keil 开发软件编写一个简单的单片机程序， 并在 ME800 系列单片机开发试验仪上进行仿真、编程和实验。

本章假设用户已经安装好了所有必需的软件， 如 Keil C51, ICE52 仿真驱动程序，USB 驱动等等， 关于软件的安装请参考本手册的第二章。

3.1 第一个 Keil C51 程序

1. 启动 Keil C51 软件。您可以通过双击电脑桌面上的"Keil uVision3"快捷方式图标来启动。
2. 执行 Keil C51 软件的菜单"Project | New Project…"，弹出一个名为"Create New Project"的对话框。先选择一个合适的文件夹准备来存放工程文件，比如"E:\Project\LedFlash"，其中"ME850Demo"是新建的文件夹。

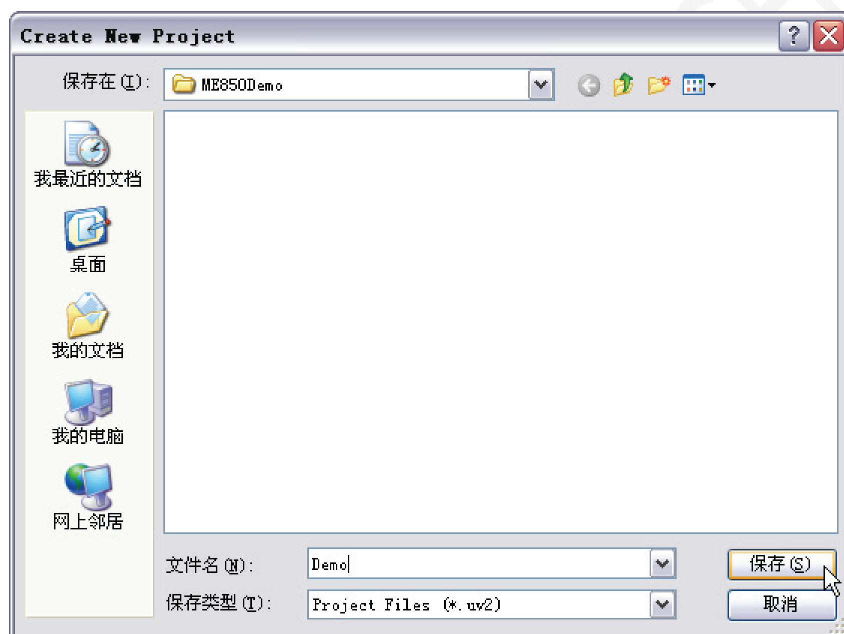


图 3.1

3. 紧接着 Keil C51 提示您选择 CPU 器件。我们选择 ATMEL 公司的 AT89S52。参见图 3.2。

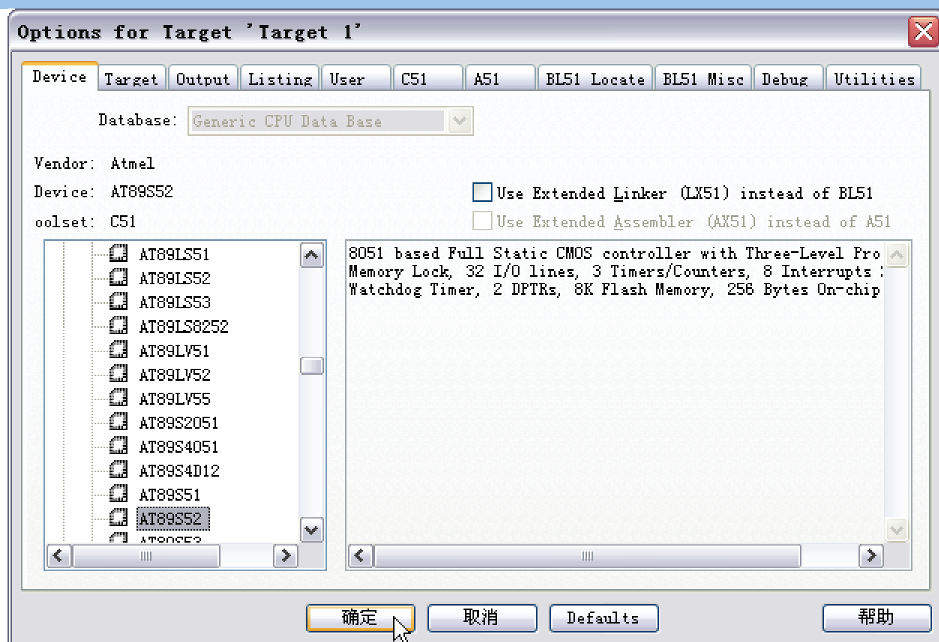


图 3.2

4. 接下来弹出一个如图 1.5 所示的对话框。该对话框提示您是否要把标准 8051 的启动代码添加到项目中去。Keil C51 既支持 C 语言编程也支持汇编语言编程。如果打算用汇编语言写程序，则应当选择" (N)"。如果打算用 C 语言写程序，一般也择" (N)"，但是，如果用到了某些增强功能需要初始化配置时，则可以选择"是(Y)"。在这里，我们选择" (N)"，即不添加启动代码。

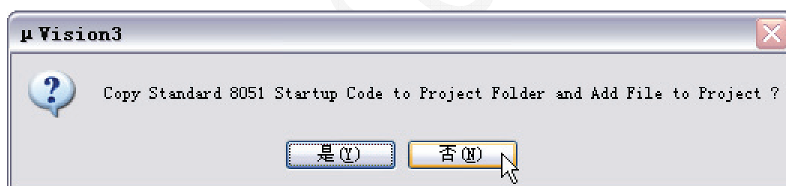


图 3.3

至此，一个空的 Keil C51 项目建立完毕。

5. 执行菜单"File | New…",出现一个名为"Text 1"的文档。 接着执行菜单"File | Save",弹出一个名为"Save As"的对话框。将文件名改为"LED.ASM"，然后保存，参见图 1.6。

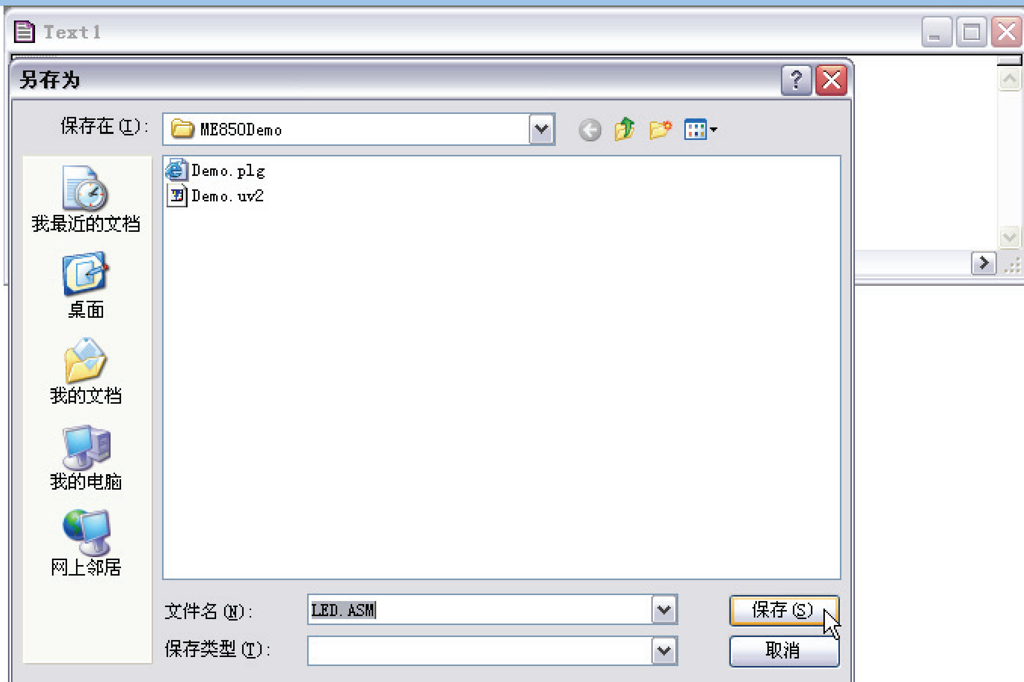


图 3.4

6. 添加源程序文件到工程中。现在，一个空的源程序文件"LED.ASM"已经建立，但是这个文件与刚才新建的工程之间并没有什么内在联系。我们需要把它添加到工程中去。单击 Keil C51 软件左边项目工作窗口"Target 1"上的" + "，将其展开。然后右击"Source Group 1"文件夹，会弹出如图 3.5 所示的选择菜单。单击其中的"Add Files to Group 'Source Group 1'"项，将弹出如图 1.8 所示的对话框。

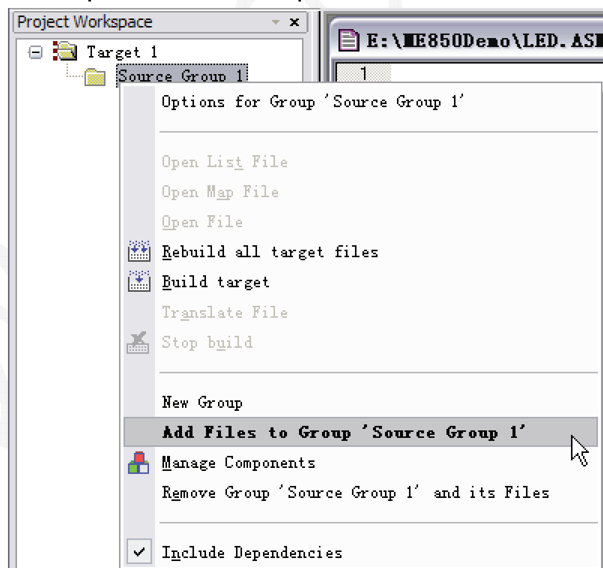


图 3.5

7. 先选择文件类型为"Asm Source file (*.s*; *.src; *.a*)"，这时，对话框内将出现刚才保存过的"LED.ASM"。单击文件"LED.ASM"，再按一次"Add"按钮，最后按"Close"按钮。这时，源程序文件"LED.ASM"已经出现在项目工作窗口的"Source Group 1"文件夹内，可以单击左边的" +"展开查看。

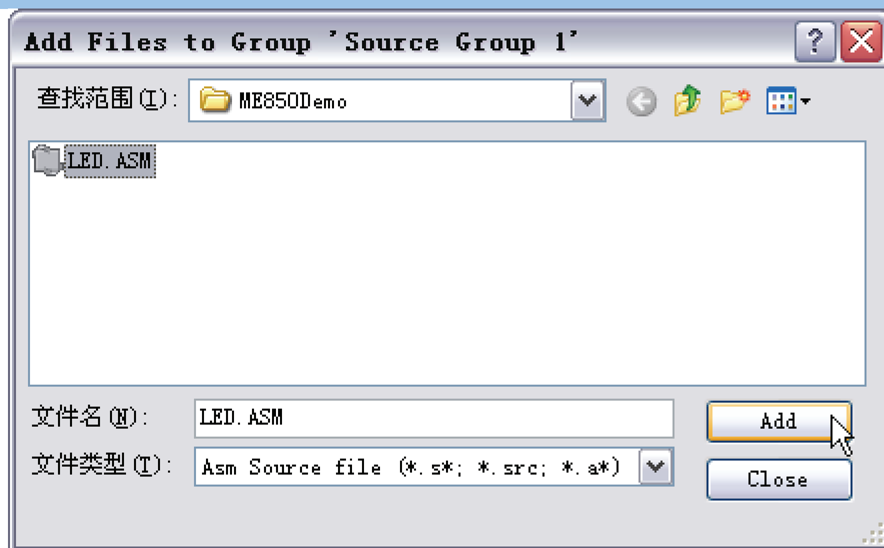


图 3.6

8. 现在开始输入源程序。先最大化"LED.ASM"源程序窗口，然后请按程序清单 1.1 输入程序代码。

```
        ORG     0000H
        LJMP    MAIN

        ORG     0050H
MAIN:
        CLR     P0.0            ; P0.0 端口低电平，点亮 D00 发光二极管
        LCALL   DELAY
        SETB    P0.0            ; P0.0 端口高电平，熄灭 D00 发光二极管
        LCALL   DELAY
        LJMP    MAIN

DELAY:                          ; 延时子程序
        MOV     R7, #250        ;
L1:
        MOV     R6, #250
L2:
        DJNZ    R6, L2
        DJNZ    R7, L1
        RET                     ; 返回主程序

        END
```

程序清单 3.1

9. 单击工具栏的按钮 编译当前源程序。编译结果会显示在输出窗口内。如果是"0 Error(s), 0 Warning(s)."，就表示程序没有问题了（至少是在语法上不存在问题了） 参见图 3.7。如果存在错误或警告，请仔细检查您的程序是否与程序清单 3.1 一致。修改后，再编译，直到通过为止。参见图 3.7。

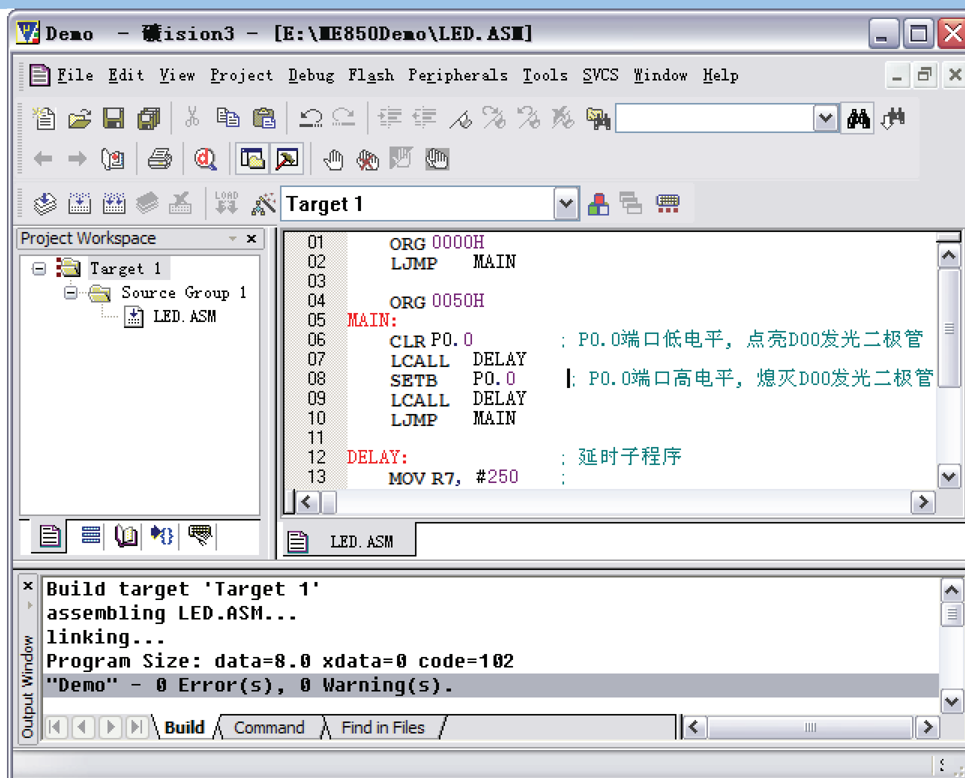


图 3.7

至此我们的第一个 Keil C51 程序已经完成．接下来的章节将讲述如何对该程序进行仿真以及将该程序下载到实验芯片中去运行．

3.2 仿真调试

1. 用随机的 10PIN 电缆将仿真器适配头（POD52）连接到 ME800 系列开发板（实验仪）的 ICE/ISP 接口。将仿真头插入到 ME800 主板的锁紧锁座上。用 USB 电缆连接好计算机与 ME800 主板。开启 ME800 的电源开关(将开关拨到 USB 一边)。如下图所示：

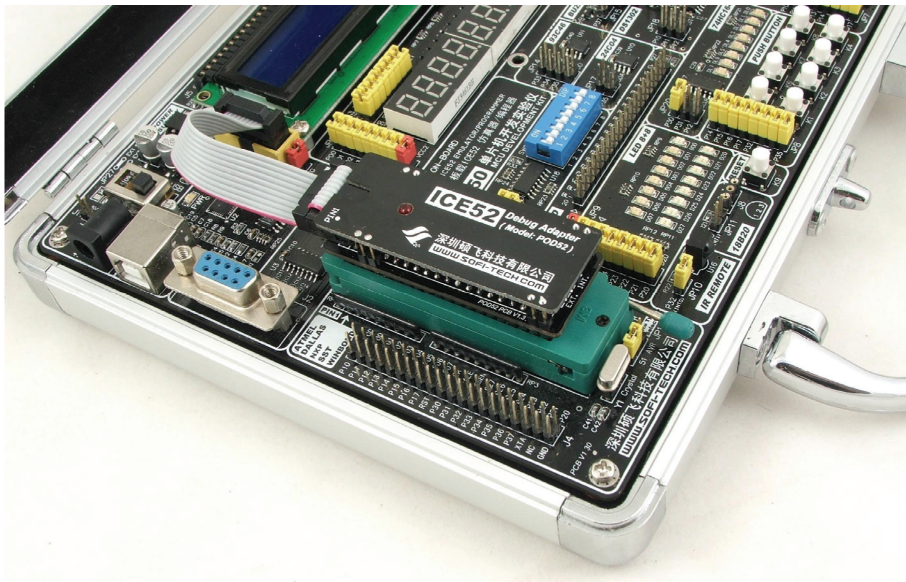


图 3.8 仿真头安装

2. 启动 Keil 软件，开启上一节中建立的 LED 闪烁演示例程。如图 3.9 所示。

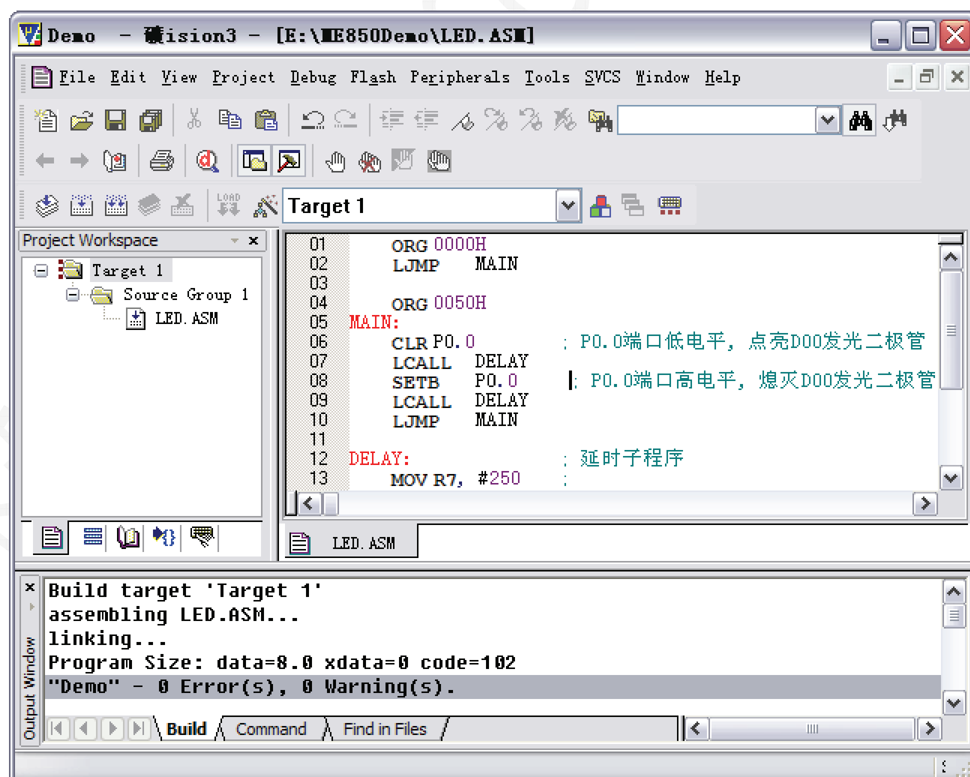


图 3.9 Keil 开发环境

3. 点击 Keil 的工具栏按钮 ，在弹出的对话框中选择"Debug"选项页。参见图 3.10 中的设置步骤进行设置。

在下拉框中如果没有发现 SOFI ICE52 Emulator/Programmer 选项，可能是没有安装 ICE52 的仿真驱动程序，请参考第二章的仿真驱动程序安装部分。其他设置请保持与图 3.10 一致.



图 3.10 Keil 仿真设置

4. 接着点击 Debug 选项页中，右上角的 Settings 按钮。软件会弹出 ICE52 的仿真/编程设置对话框，如图 3.4 所示

对于 ICE52 的仿真部分只有唯一的一个选项，即是否开启运行中停止功能。当开启该功能之后，用户不可以使用代码区域中 0x003b 处的三个字节，这个三个字节需要被仿真器占用。

对于汇编语言可以在程序开始时，请将主程序设在 003EH 之后， 如在 0050H 开始主程序。
对于 C 语言，请在源文件中加入如下代码行。
光盘中的所有示例程序已经对 0x003b 处做了保留。因此我们可以勾选该功能。



图 3.11 ICE52 设置

5. 进行完以上设置后，编译该项目，检测编译结果。编译无误后，便可进行仿真功能操作。点击 Keil 的工具栏按钮 ，开始进入仿真状态, Keil 的界面显示如图 3.12 所示。点击 Debug 菜单下的仿真命令或者相应的工具栏按钮，即可进行仿真操作．包括全速运行，单步运行，跨步运行，断点的设置/取消等等。

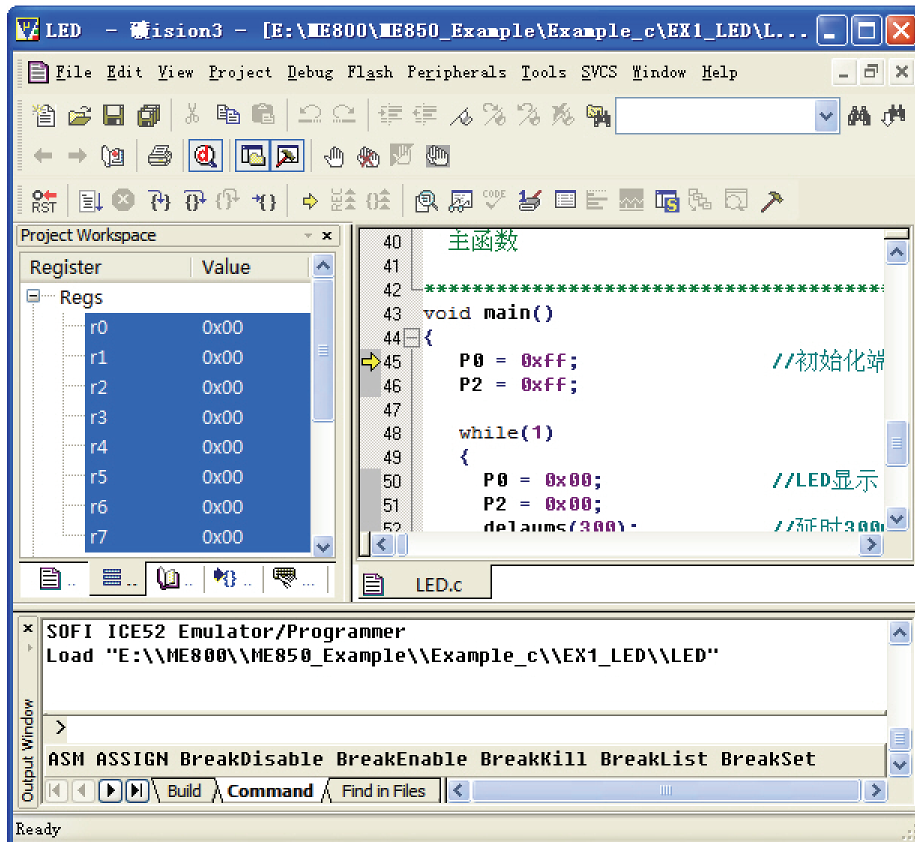


图 3.12　仿真界面

3.3 下载运行

1. 点击工具栏按钮 ，然后点击 Unitilies 选项页，按图 3.13 中的设置步骤进行设置。

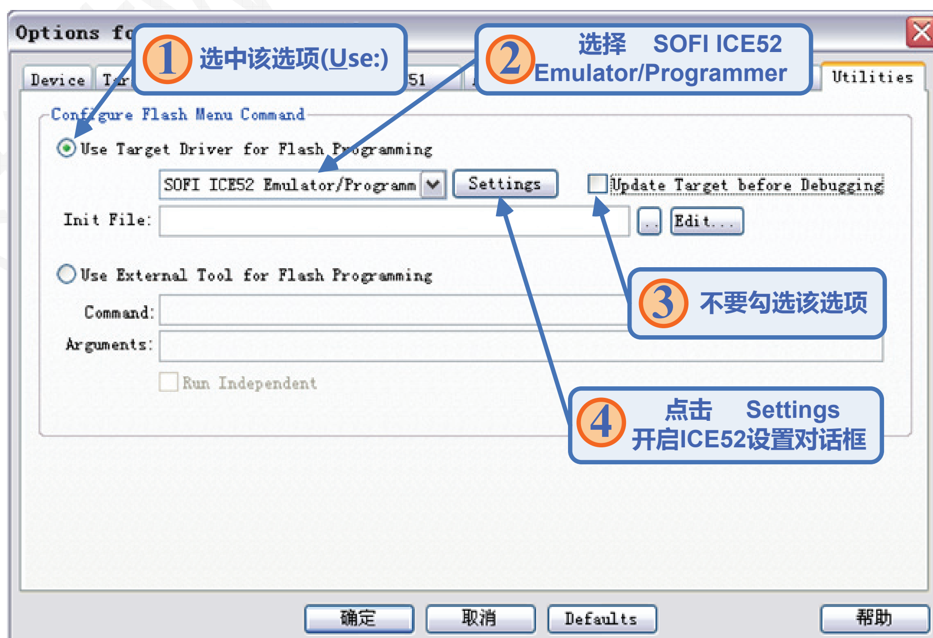


图 3.13　Keil 编程设置

2. 选中 Use Target Driver for Flash Programming 选项，在列表框中选择 SOFI ICE52 Emulator/Programmer。然后点击Settings按钮，打开ICE52的仿真/编程设置对话框，如下图(图3.14)所示。在对话框的下载编程选项中分别选择合适的器件厂商和芯片型号。



图 3.14 ICE52 设置

3. 将产品附带的 AT89S52 芯片放置到 ME8 5 0 的锁紧座上，如图 3.15 所示.

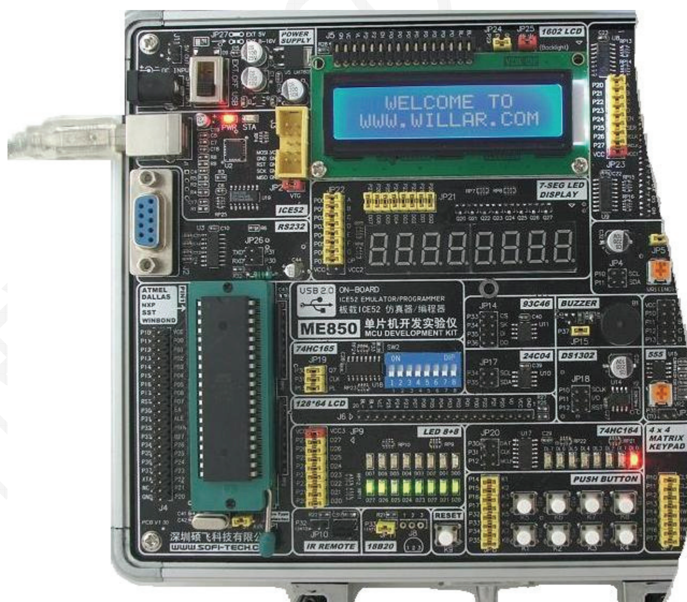


图 3.15 实验芯片放置

4. 点击 Keil 工具栏的点击 Keil 工具栏的按钮，或者是菜单【Flash】→【Download】即可将程序烧写开发板的实验芯片(AT89S52)内。Keil 的窗口内会显示如图 3.16 所示的内容. 此时开发板上单片机便开始运行之前该程序.

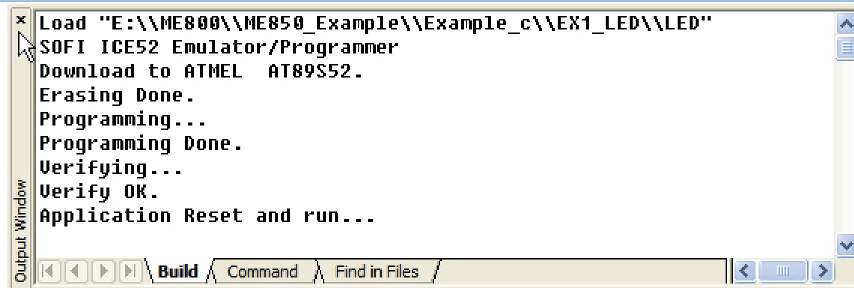


图 3.16 Keil 编程信息

第四章 仿真器/编程器 ICE52

开发实验仪上内嵌了 USB2.0 接口的仿真器/编程器 ICE52．该仿真器能够与目前最专业 Keil 开发环境使用，在仿真过程中不会占用用户的任何资源，特别是不占用 IO 端口，以及串口和定时器等重要资源．

仿真器搭配独立的 POD52 仿真适配头，不仅可以仿真标准 8051/8052 单片机的所有功能．还可以仿真部分单片机的增强型功能．该仿真适配头采用 CPU 直接置入方式，将仿真的 CPU 频率直接提高到其极限，并且稳定可靠．

编程器功能不仅可以使用 MEFlash 进行操作，还可以在 Keil 中直接进行烧写，当程序编译完成后，仅需要点击一下 Keil 功能栏的 FLASH DOWNLOAD 按钮，即可将代码快速下载到单片机中．

4.1 功能特点

- USB2.0 接口(真正的 USB 接口，非 USB 转串口)增强型仿真器
- 集成编程下载功能，支持 Keil 的下载编程操作
- 可仿真 ATMEL, WINBOND, DALLAS, INTEL, SST, PHILIPS 等所有兼容 51 与 52 单片机
- 不占用用户资源，完全真实单片机所有端口特性
- 快速代码下载，极速单步操作
- 全新开发的 Keil 仿真接口驱动，与开发环境完美接合
- 支持软件复位操作，无需手动复位硬件，即可进行连续代码下载与仿真操作
- 支持脱机运行
- 微型置入式仿真适配头（POD52），彻底避免传统仿真排线带来的多种不稳定因素
- 支持标准的仿真操作，如全速，单步，跨步运行，断点的设置/禁用/取消，寄存器与变量查看
- 支持运行中暂停(夭折)功能
- 可仿真双 DPTR, PCA, ALE 禁用, SPI 接口，片内 768 字节扩展 RAM 等增强型 51 单片机资源
- 高达 63K 的代码仿真空间，支持外部 64K 扩展 RAM 仿真

4.2 仿真功能

在安装完所需的 Keil 软件已经 ICE52 仿真驱动程序之后（请参考本手册第二章相关章节）， 在 Keil 项目中还需进行相关设置，才能使用仿真功能。请参考本手册第三章的有关说明来设置仿真功能。

■ 仿真适配头的安装

用随机附带 10pin 连接线，一端插入开发板的 J3 接口，另一端插入 POD52 仿真适配头．然后将仿真头放置到开发板的实验芯片座上．参见图 4.1。

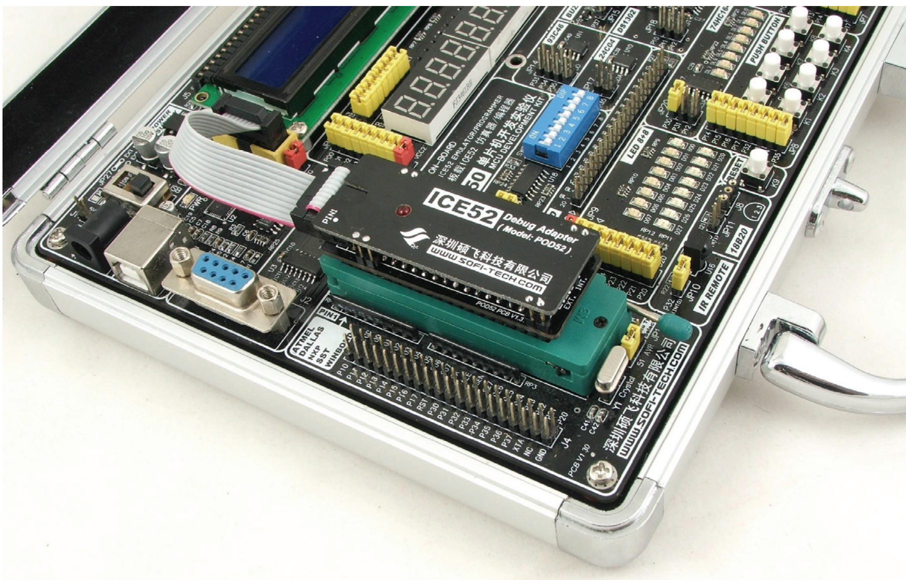


图 4.1 仿真头安装

仿真头 POD52 上有一个晶振选择开关，用于选择 POD52 是使用内置晶振还是用户板上的晶振。将开关设置到 INT 选择 POD52 内置的 11.0592MHz 晶振。拨到 EXT 端选择开发板上或者是用户板上的晶振。

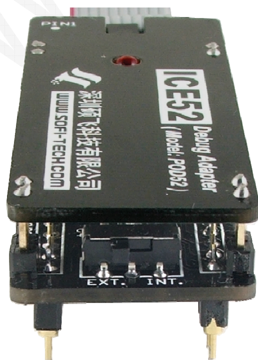


图 4.2 仿真头晶振设置

■ 断点设置与取消

ICE52 最大可以支持 19 个固定断点(即地址断点)和一个临时断点(用于跨步运行)。断点的设置与取消操作比较简单，仅需要将光标移到相应的源代码行，然后点击的 Debug 菜单中的 Insert/Remove Breakpoint 或者按一下键盘的 F9 键即可。在断点使能后，在当前源代码行的前面会显示一个红色的方块标志，如下图所示.

```
039        MOV      P0, #0BFH
040        LCALL    DELAY
041        MOV      P0, #07FH
```

图 4.3 断点设置

用户设置的地址断点不可以超过 19 个，如果断点超过该限制，在开始全速运行或跨步运行时，Keil 会在 Command 窗口中显示如下图所示的提示信息:

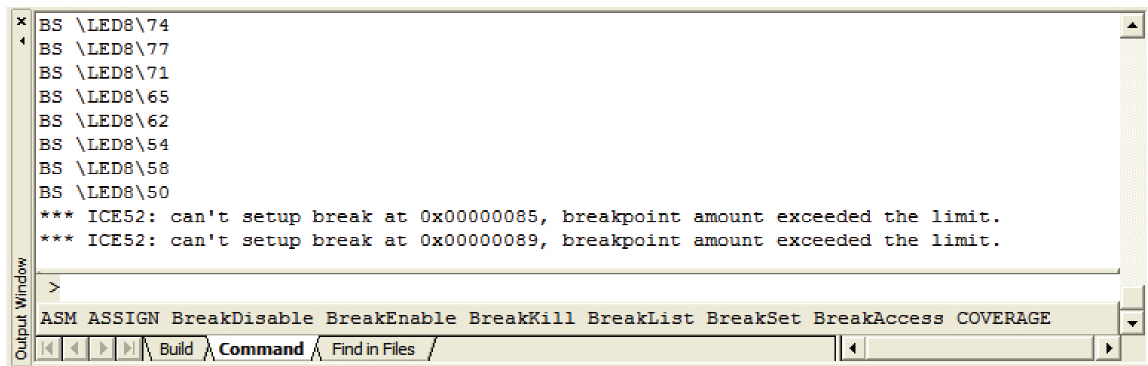


图 4.4 错误提示

超过数量的断点将自动被禁用，禁用的断点会显示成灰白色，如下图所示:



图 4.5 禁用的断点

仿真运行

进入仿真模式之后，可以通过菜单 Debug 已经相应的工具栏按钮执行相应的运行操作。

Debug 菜单	工具栏按钮	功能
Run		全速运行
Step		单步运行
Step Over		跨步运行
Run to Cursor line		运行到光标所在行
	RST	复位
Stop Running		暂停(需开启暂停功能)

暂停功能

在仿真器全速运行过程中，可以点击 Keil 工具栏的按钮 ，或者是菜单【Debug】→【Stop Running】将仿真器暂停下载，暂停后可以再次点击全速/单步运行和跨步运行。

此功能是一个可选项，需在 ICE52 对话框中进行设置，只有勾选此选项后，该功能才有效。

开启该功能后， 仿真器需要占用代码空间的 003BH 的三字节空间，用户的代码不可以使用该区域。对于 C 语言程序可以使用如下的代码将该空间进行保留。

```
char code reserve [3] _at_ 0x3b;          // 保留三字节
```

对于汇编语言， 可以直接将主程序跳过该区域， 如将主程序定位在 0050H 之后，代码如下：

```
        ORG     0000H
        LJMP    MAIN

        ORG     0050H
MAIN:
        ;主程序开始
```

该功能借助 51 的单片机的内部的中断功能实现，所以在用户的代码中不可以关闭中断功能（例如将 EA 设为了 0），否则仿真器运行后，将无法暂停。软件会显示如下所示的提示:

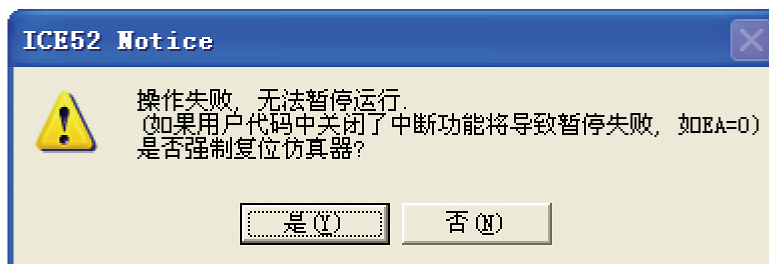


图 4.6 ICE52 Notice

■ 仿真扩展 RAM

仿真监控 CPU 是一个增强型的 51 单片机，片内已带有 768 字节的扩展 RAM，在默认模式(上电复位后)，MOVX 指令对地址范围 000H~2FFH 的读写操作，将直接访问片内的扩展 RAM，当地址超过 2FFH 时，才会访问片外扩展的 RAM。

如果需要直接访问片外 RAM，可以将片内的扩展 RAM 禁用，通过设置 AUXR 寄存器的 EXTRAM 位为 1 即可。

AUXR 寄存器 EXTRAM 位	MOVX 访问地址 00H~2FFH	MOVX 访问地址 300H~FFFFH
EXTRAM=0 (芯片默认模式)	访问片内扩展 RAM	访问片外扩展 RAM
EXTRAM=1	访问片外扩展到 RAM	

■ 仿真外部目标板

ME800 系列开发板搭载的 ICE52 仿真器支持对外部目标板的仿真。直接将仿真头插入到外部目标板的 CPU 插座上即可，如下图所示。

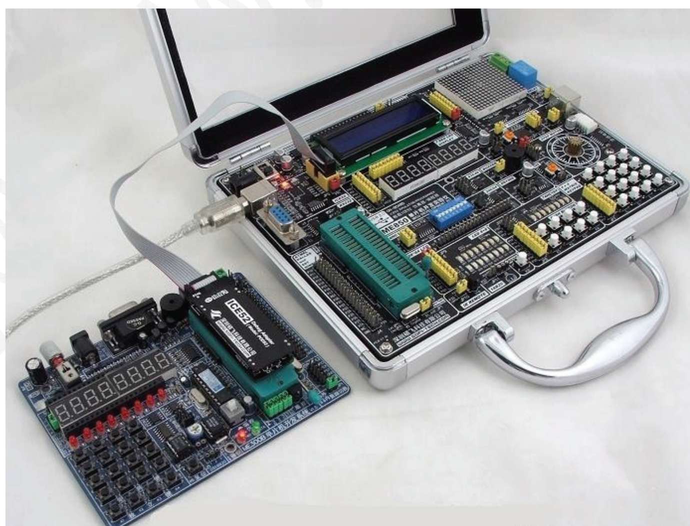


图 4.7 仿真外部目标板

对外部仿真时，短接 ME800 系列开发实验仪上 JP2(VTG)跳线，可对外部目标板进行供电。

■ 脱机运行

POD52 仿真头支持脱机运行，直接将 POD52 仿真头插入目标板的 CPU 插座，仿真头便会自动运行上一次的调试过的程序.

脱机运行的程序也可以通过 Keil 的 Flash Download 功能重新下载. 在 ICE52 的设置对话框中的下载编

程设置区内选中"下载到仿真适配头（POD52)"，然后执行 Keil 的 Flash Download 命令，Keil 自动将程序下载到仿真适配头的 CPU 内，然后开始全速运行．运行效果与一个真实的 CPU 完全相同．

4.3 编程功能

ICE52 支持编程功能，只需要在项目编译通过后，点击 Keil 工具栏的 LOAD 按钮，便可将代码下载到芯片中运行，非常方便快捷。编程操作步骤请参考第二章的编程快速入门.

■ 器件列表

器件厂商	器 件 型 号
ATMEL	AT89S51 AT89S52 AT89S53 AT89S8252 AT89S51@ISP AT89S52@ISP AT89S53@ISP AT89S8252@ISP ATmega8515 ATmega8525L ATmega8515@ISP ATmega8515@ISP ATmega8@ISP ATmega16@ISP ATmega32@ISP ATmega64@ISP ATmega128@ISP ATmega48@ISP ATmega88@ISP ATmega168@ISP ATmega8L@ISP ATmega16L@ISP ATmega32L@ISP ATmega64L@ISP ATmega128L@ISP ATmega48L@ISP ATmega88L@ISP ATmega168L@ISP ATmega48V@ISP ATmega88V@ISP ATmega168V@ISP
DALLAS(MAXIM)	DS89C430 DS89C440 DS89C450
NXP(Philips)	P89V51RB2 P89V51RC2 P89V51RD2
SST	SST89E52 SST89E54 SST89E58 SST89E516 SST89E564
WINBOND	W78E58B W78E516B

说明:

- ✧ 上表中的灰色部分的器件为 AVR 系列的单片机，该部分芯片不能在 Keil 中进行下载(Keil 仅支持 51 系列单片机)，必须使用 MEFlash 专业编程软件来进行烧写，详情请查看 MEFlash 软件使用手册。
- ✧ 带@ISP 后缀的器件采用 ME800 主板上的 ISP 接口进行下载(即外部 ISP 编程) ,不能直接放在主板的锁紧座上烧写。同样 ,不带@ISP 后缀的器件只能放在主板的锁紧上烧写 ,不可以通过 ISP 接口进行烧写。

■ 外部 ISP 编程

ME800 系列开发板(实验仪)上搭载的 ICE52 仿真编程器支持对用户板进行 ISP 编程操作，在 ME800 的主板上有个 ICE/ISP 连接端口，通过产品附带的 10Pin 连接线将此端口与用户的目标板进行连接即可实现对用户板上的单片机进行下载编程操作。

ME800 系列板上的端口引出方式如图 5.2 所示.

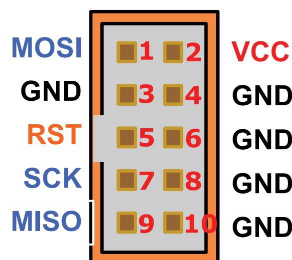


图 5.2 ISP 编程接口

对外部 AT89S51/S52/S53/S8252 等芯片进行烧写的连接方式如下:

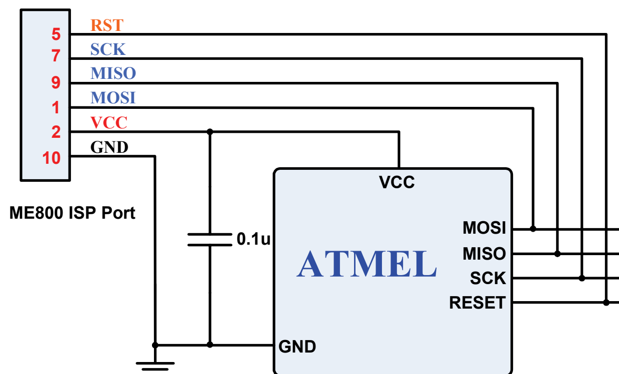


图 5.3 ATMEL 51 单片机 ISP 连接方式

对外部 AVR 系列芯片的连接方式如下:

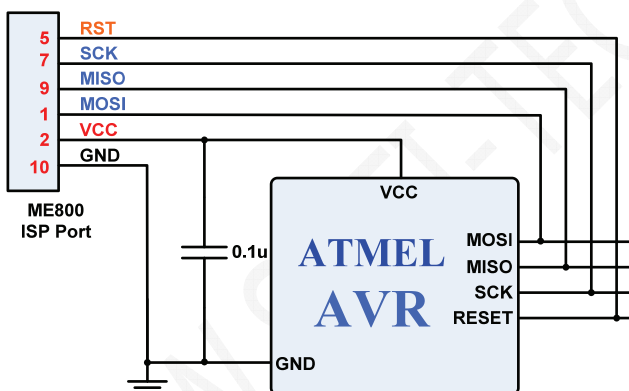


图 5.4 AVR ISP 连接方式

ME850 对外部进行 ISP 编程示意图:

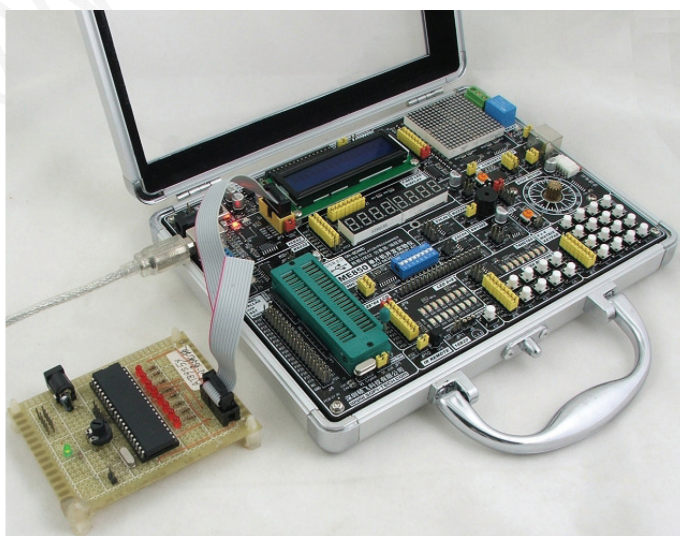


图 5.5 外部 ISP 编程

说明:

外部 ISP 编程仅支持器件列表中带@ISP 后缀的单片机

器件编程特殊说明

ME800 系列开发板(实验仪)支持多个厂商的单片机，所有的单片机均都使用其 ISP 功能进行烧录和下载。所以在 ME800 上使用这些芯片必需保证其 ISP 功能正常。部分芯片的 ISP 功能是固有的，例如 ATMEL 公司的 AT89S51/S52，这些芯片在 ME800 上可以直接使用，无需任何预先设置。但是有些芯片的 ISP 功能可能需要进行预先设置，才能保证在 ME800 上正确使用。

- **ATMEL 单片机**

ATMEL 单片机主要指 AT89S51/S52/S53/S8252/S8253 等芯片，该系列即可放在 ME800 系列的主板锁紧座上进行烧写，也可以通过 ICE52 的 ISP 接口对用户的目标板进行下载(即外部 ISP 编程)。

无论是放在锁紧座上，还是对用户的目标板进行下载，芯片都必须外接晶振，范围 3~24MHz。ME800 系列主板上默认的晶振是 11.0592MHz(出厂配置，用户可以根据需要更换)

在对用户目标板的芯片进行下载时，还必须保证芯片的有可靠的供电，供电可以是用户的目标板自供电也可以通过 ME800 的 ISP 接口进行供电。在使用 ME800 的 ISP 接口进行供电时，需短接 ME800 主板上的 JP2 跳线。

- **SST 单片机**

该系列单片机包含两个程序存储区，分别为 BLOCK0 用户程序区，和 BLOCK1 启动代码程序区。要在 ME800 系列开发板(实验仪)上实现此类芯片的编程下载操作，BLOCK1 必须内置有 ISP 启动下载代码，并且单片机的配置位 CS0/CS1(SST89E516 和 SST89E516 只有 CS0)必须进行相应的设置，使单片机复位后从 BLOCK1 启动。

该芯片在出厂时，通常已经烧录有 BSL 并配置好相应的配置位。即芯片如果是全新没有使用过的，可以直接在开发板(实验仪)上使用。

如果是已经使用过的芯片，其 BSL 和配置位很有可能已经被破坏，此时必须使用其他商用编程器进行恢复。产品光盘中有该 BSL 的代码文件，位于 BOOTLOADER\SST\目录下，文件名为 F51MBLL5。BIN。将该代码烧录到 BLOCK1 区，然后设定正确的配置位即可。设置方式请参考下表

BSL 代码及配置位设置表

芯片型号	BSL 文件	缓冲区开始地址 (BLOCK1 映射地址)	配置位设置
SST89E52	F51MBLL5.BIN	0xE000	SC0=1，CS1=1
SST89E54	F51MBLL5.BIN	0xE000	SC0=1，CS1=1
SST89E58	F51MBLL5.BIN	0xE000	SC0=1，CS1=1
SST89E516/564	F51MBLL5.BIN	0x10000	SC0=1

说明:

- ✧ 在使用其他编程器加载 FM51LL11.BIN 文件时，需要设置缓存区开始地址，即上表中的 BLOCK1 映射地址。
- ✧ 配置位未编程时(即擦除后)等于 1，对该配置位执行编程操作后等于 0。
- ✧ 该系列芯片在出厂时已烧录好 BSL 和相应的配置位，可以直接在开发板(实验仪)上使用。
- ✧ ME800 系列开发板(实验仪)只能实现对 BLOCK0 区的烧写，不可以烧写 BLOCK1。

- **WINBOND 单片机**

此类芯片与 SST 的单片机类似，也包含两个存储区。一个为 APROM，一个为 LDROM。APROM 为用户代码存储区，LDROM 是启动代码存储区。LDROM 必须预先烧录相应的启动下载代码，才可以在 ME800 系列开发板(实验仪)上使用。

WINBOND 单片机的启动代码位于产品光盘的 BOOTLOADER\WINBOND\目录下，文件名为 WB_ISP。HEX。烧录该文件必须使用其他的商业并行编程器。在加载该代码时，请注意缓冲区的偏移量设置，必须保证 WB_ISP。hex 正确的调入到 LDROM 开启区。

第五章 实验指导

5.1 基础实验

实验一 LED 闪烁

发光二极管是半导体二极管的一种，可以把电能转化成光能。常简写为 LED (light emitting diode)。

发光二极管与普通二极管一样也具有单向导电性。当给发光二极管加上正向电压（大于 LED 的正向压降）就会发光，当给发光二极管加上负向电压就不会发光。

发光二极管的发光亮度与通过的工作电流成正比，一般情况下，LED 的正向工作电流在 10mA 左右，若电流过大时会损坏 LED，因此使用时必须串联限流电阻以控制通过管子的电流。限流电阻 R 可用下式计算：

$$R = (E - UF) / IF$$

式中 E 为电源电压，UF 为 LED 的正向压降，IF 为 LED 的一般工作电流。

普通发光二极管的正向饱和压降为 1.4～2.1V,正向工作电流为 5～20mA.

LED 广泛应用于各种电子电路、家电、仪表等设备中、做电源或电平指示。

1. 实验任务

P0、P2 端口的 LED 亮 300ms，灭 300ms，如此循环。

发光二极管在不停地一亮一灭，间隔时间为 300ms，形成闪烁报警效果。

2. 实验电路

2.1 实验 CPU 电路

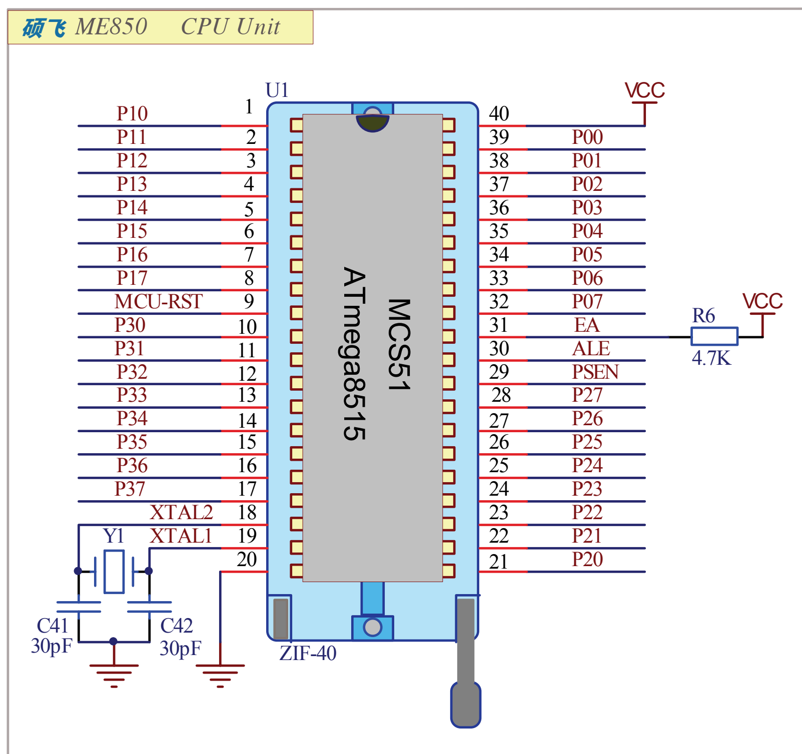


图 5.1 实验 CPU 电路

实验 CPU 插在 40Pin 活动锁紧座 U1 上（图 5.1），可方便更换不同型号的 51 或 AVR 单片机进行烧写或实验，芯片在锁紧座上的方向是芯片缺口背向手柄。注意：实验插座上只能放置和 89C51 管脚兼容的 51 系列单片机（如 AT89S521/52），同样只能放置和 8515 管脚兼容的 AVR 系列单片机（如 ATmega8515L）。

Y1 是可以更换的晶振。

实验 CPU 的复位电平"MCU-RST"来自系统控制芯片，兼容 51、AVR 单片机的正常复位，按下实验仪上的复位键 K9，实验 CPU 即得到复位信号。

注意：本手册原理图和 ME850 电路板中 CPU 的 I/O 口标识省掉了小数点，如 P10 表示实际端口 P1.0，P20 表示实际端口 P2.0。

2.2 LED 显示电路

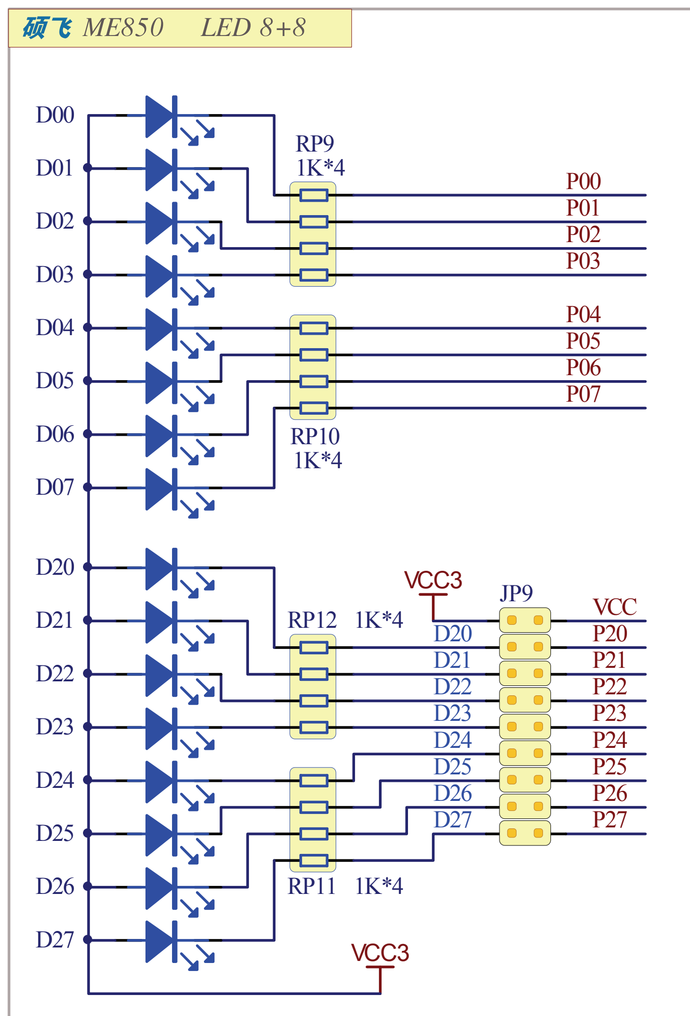


图 5.2 LED 显示电路

RP9、RP10、RP11、RP12 为发光二极管的限流电阻。

用 P0 和 P2 端口作输出口，每个端口接 8 位用作逻辑电平显示的发光二极管。

D00～D07 为发红色光的发光二极管，D20～D27 为发绿色光的发光二极管。

当 P0 和 P2 端口输出低电平时，D00～D07 和 D20～D27 正向导通发光。

当 P0 和 P2 端口输出高电平时，D00～D07 和 D20～D27 截止不发光。

3. 实验步骤

将 JP9 的 9 个短接子全部用短接帽短接，使 D20～D27 与 P2 端口接通，VCC 向发光二极管模块供电；

1602LCD 上端的 JP24 短接在 OFF 端，避免 1602 干扰 LED。

4. 程序流程图

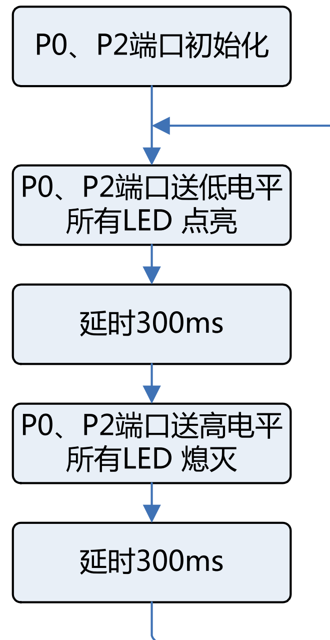


图 5.3 EX1_LED 流程图

5. 汇编源程序

（Example_A51\EX1_LED）

```
        ORG   0000H
        AJMP   MAIN
        ORG   0050H

MAIN:
        MOV   P0,#0FFH          ;端口初始化
        MOV   P2,#0FFH
LOOP:
        MOV   P0,#00H           ;LED 显示
        MOV   P2,#00H
        ACALL   DELAY          ;延时 300ms

        MOV   P0,#0FFH          ;关闭 LED 显示
        MOV   P2,#0FFH
        ACALL   DELAY          ;延时 300ms

        AJMP   LOOP

;---------------------------------------------
; 延时子程序
; 延时 300ms (11.0592MHz)
;---------------------------------------------
DELAY:
        MOV   R5,#3
DEL1:
        MOV   R6,#200
DEL2:
        MOV   R7,#230
DEL3:
        DJNZ   R7,DEL3          ;第一层循环
```

```
        DJNZ   R6,DEL2          ;第二层循环
        DJNZ   R5,DEL1          ;第三层循环
        RET

        END                      ;结束
```

6. C 语言源程序

（Example_C51\EX1_LED）

```
#include <reg52.h>
char code reserve [3] _at_ 0x3b;   //保留 0x3b 开始的 3 个字节

/*-------------------------------------------------------
  延时函数
-------------------------------------------------------*/
void delayms(unsigned int ms)
{
  unsigned char k;
  while(ms--)
  {
    for(k = 0; k < 114; k++);
  }
}

/*-------------------------------------------------------
  主函数
-------------------------------------------------------*/
void main()
{
   P0 = 0xff;                  //初始化端口
   P2 = 0xff;

   while(1)
   {
     P0 = 0x00;                //LED 显示
     P2 = 0x00;
     delayms(300);             //延时 300ms

     P0 = 0xff;                //关闭 LED 显示
     P2 = 0xff;
     delayms(300);             //延时 300ms
   }
}
```

实验二 LED 流水灯

1. 实验任务

P0、P2 端口的 LED 先从从右至左方向依次点亮，再从左至右方向依次点亮，如此循环形成流水灯效果。

2. 实验线路 （见图 5.1）

3. 实验步骤

将 JP9 的 9 个短接子全部用短接帽短接，使 D20～D27 与 P2 端口接通，VCC 向发光二极管模块供电；1602LCD 上端的 JP24 短接在 OFF 端，避免 1602 干扰 LED。

4. 程序流程图

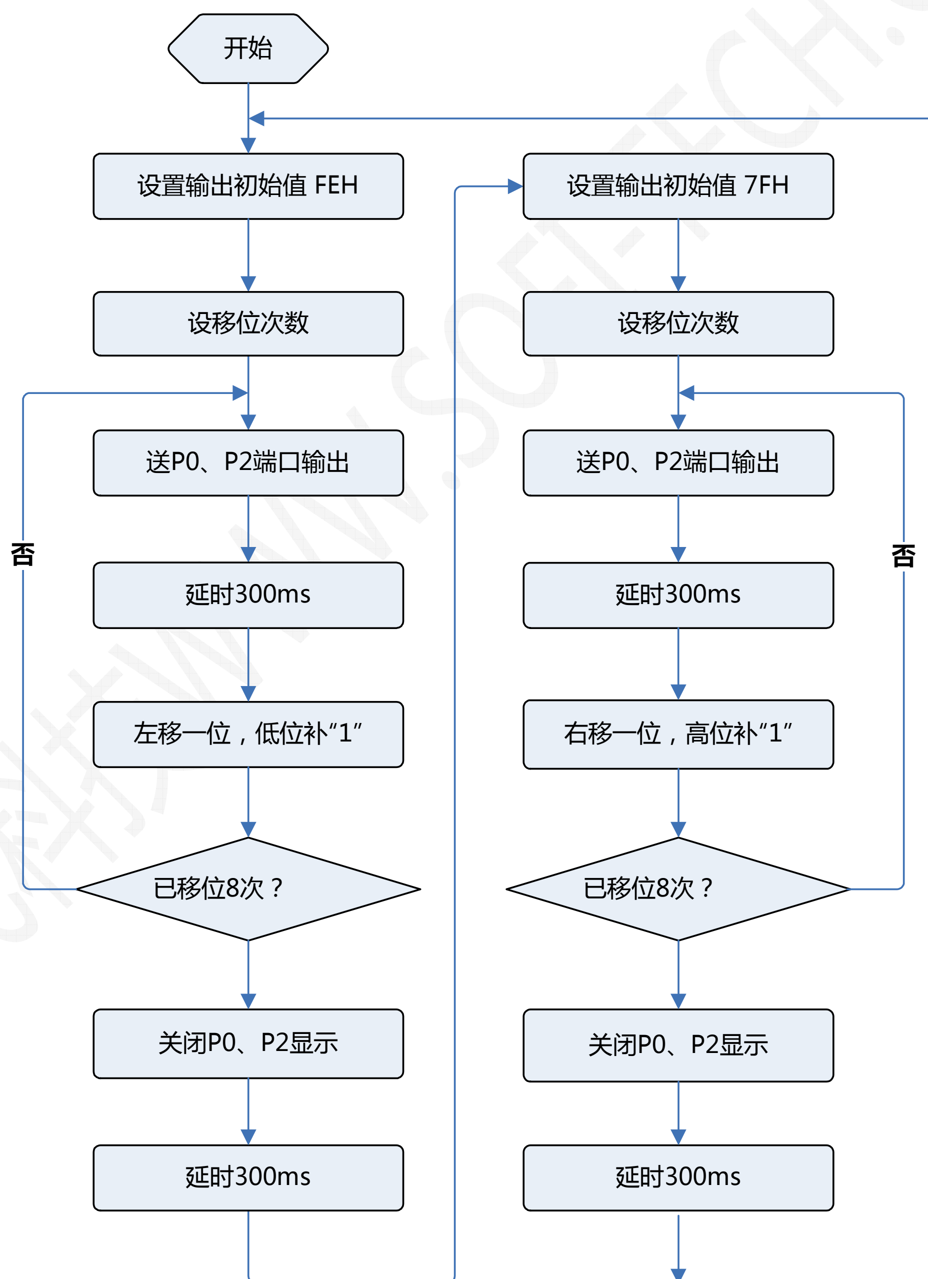


图 5.4 EX2_LEDX8 流程图

5. 汇编源程序

（光盘： Example_A51\EX2_LEDX8）

```
        ORG   0000H
        AJMP  MAIN
        ORG   0050H

;-----------------------------------------------
MAIN:
        MOV   P0,#0FFH          ;端口初始化
        MOV   P2,#0FFH
LOOP:
        MOV   A,#0FEH           ;赋初始值
        MOV   R0,#08H           ;移动次数
LOOPL:                          ;左移显示
        MOV   P0,A              ;送数显示
        MOV   P2,A
        RL    A                 ;左移一位
        ACALL   DELAY           ;延时 300ms
        DJNZ   R0,LOOPL         ;是否左移 8 次？

        MOV   P0,#0FFH          ;关闭显示
        MOV   P2,#0FFH
        ACALL   DELAY           ;延时 300ms

        MOV   A,#7FH            ;赋初始值
        MOV   R0,#08H           ;移动次数
LOOPR:                          ;右移显示
        MOV   P0,A              ;送数显示
        MOV   P2,A
        RR    A                 ;右移一位
        ACALL   DELAY           ;延时 300ms
        DJNZ   R0,LOOPR         ;是否右移 8 次？

        MOV   P0,#0FFH          ;关闭显示
        MOV   P2,#0FFH
        ACALL   DELAY           ;延时 300ms

        AJMP   LOOP

;-----------------------------------------------
; 延时子程序
; 延时 300ms (11.0592MHz)
;-----------------------------------------------
DELAY:
        MOV   R5,#3
DEL1:
        MOV   R6,#200
DEL2:
        MOV   R7,#230
DEL3:
        DJNZ   R7,DEL3          ;第一层循环
        DJNZ   R6,DEL2          ;第二层循环
        DJNZ   R5,DEL1          ;第三层循环
        RET
;-----------------------------------------------
```

```
        END               ;结束
;---------------------------------------------
```

6. C 语言源程序

（光盘： Example_C51\EX2_LEDX8）

```
#include <reg52.h>
#include <intrins.h>

unsigned char  scan_num;

char code reserve [3] _at_ 0x3b;  //保留 0x3b 开始的 3 个字节

/*-------------------------------------------------------

  延时函数

-------------------------------------------------------*/
void delayms(unsigned int ms)
{
  unsigned char k;
  while(ms--)
  {
    for(k = 0; k < 114; k++);
  }
}

/*-------------------------------------------------------

  主函数

-------------------------------------------------------*/
void main(void)
{
   unsigned char i;
   P0 = 0xff;                  //初始化端口
   P2 = 0xff;

   while(1)
   {
     scan_num = 0xfe;           //扫描初始值

     for(i = 0;i < 8;i++)      //左移显示
      {
       P0 = scan_num;           //送显示
       P2 = scan_num;
       scan_num<<=1;             //左移一位
       scan_num|=0x01;          //最低位补"1"
       delayms(300);            //延时 300ms
     }
     P0 = 0xff;                 //关闭 LED 显示
     P2 = 0xff;
     delayms(300);              //延时 300ms

     scan_num = 0x7f;           //扫描初始值
```

```
        for(i = 0;i < 8;i++)      //右移显示
        {
            P0 = scan_num;           //送显示
           P2 = scan_num;
           scan_num >>=1;            //右移一位
           scan_num|=0x80;          //最高位补"1"
           delayms(300);           //延时 300ms
        }
        P0 = 0xff;                 //关闭 LED 显示
        P2 = 0xff;
        delayms(300);              //延时 300ms
    }
}
```

实验三　继电器控制

继电器是一种电子控制器件，它具有控制系统（输入回路）和被控制系统（输出回路），通常应用于自动控制电路中，它实际上是用较小的电流去控制较大电流的一种"自动开关"。

1．实验任务

用按键控制继电器的工作状态：

K1- 吸合键，K2- 释放键

按 K1 键，继电器吸合，DL11 灯亮。

按 K2 键，继电器释放，DL11 灯灭。

2．实验线路

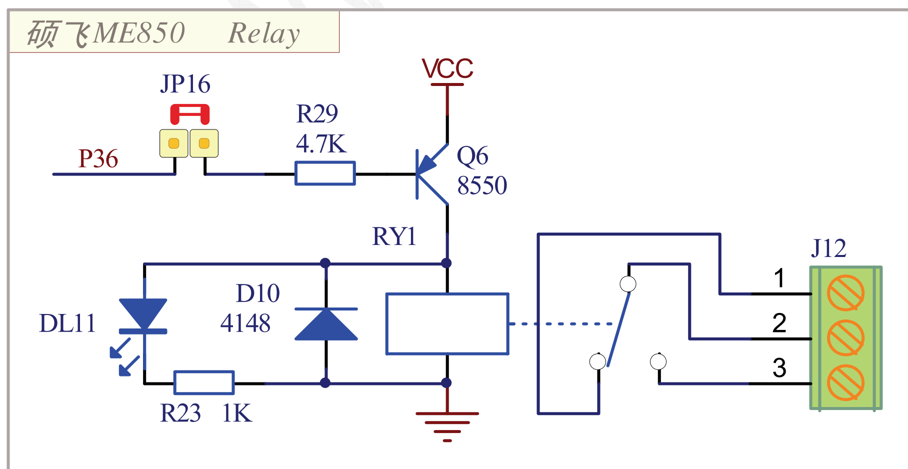


图 5.5　继电器驱动电路

因为继电器稳定吸合时，线圈的工作电流已超出 C51 芯片 I/O 端口的驱动能力，所以考虑使用三极管来驱动继电器。因继电器线圈是电感性负载，在驱动电路中还要加相应的保护措施才行。

实际的继电器控制电路如图 5.5 所示，当 P36 引脚输出"0"时，三极管 Q6 导通，继电器 RY1 吸合，发光二极管 DL11 点亮；当 P36 引脚输出"1"时，三极管 Q6 截止，继电器 RY1 释放，发光二极管 DL11 熄灭。

3. 实验步骤

- 短接 JP16 短接子，使继电器接口电路使能；
- 将 JP8 的 8 个短接子全部用短接帽短接，使独立按键与相应的端口接通。

4. 程序流程图

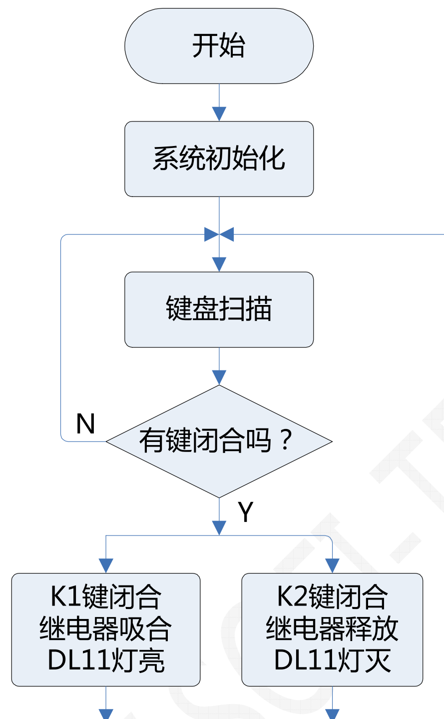


图 5.6 EX3_RELAY 流程图

5. 汇编源程序

(Example_A51\EX3_RELAY)

```
KEY_NEW     EQU  40H
KEY_OLD     EQU  41H

K1          BIT  P1.4            ;
K2          BIT  P1.5            ;
RELAY       BIT  P3.6            ;继电器控制线

;----------------------------------------------------------

        ORG   0000H
        AJMP  MAIN

        ORG   0050H
;----------------------------------------------------------

; 主程序

;----------------------------------------------------------
MAIN:
        MOV   SP,#60H          ;设置栈指针
        MOV   P0,#0FFH         ;
```

```
        MOV   P2,#0FFH              ;
        MOV   KEY_OLD,#03H           ;初始键比较值

KEY_CHK:                            ;循环检测按键是否按下
        ACALL SCAN_KEY               ;输入按键状态
        XRL   A,KEY_OLD              ;查按键值是否改变
        JZ    KEY_CHK               ;若无键被按,则跳回 KEY_CHK

        ACALL   DELAY               ;延时去抖
        ACALL   SCAN_KEY             ;再次检查按键值
        XRL   A,KEY_OLD
        JZ    KEY_CHK

        MOV   KEY_OLD,KEY_NEW        ;保存按键状态
        ACALL PROC_KEY
        AJMP   KEY_CHK

;-------------------------------------------------------

; 扫描按键子程序

; 返回值:    A --- 按键状态

;-------------------------------------------------------
SCAN_KEY:
        CLR   A
        MOV   C,K1
        MOV   ACC.0,C
        MOV   C,K2
        MOV   ACC.1,C
        MOV   KEY_NEW,A              ;无键按下 key_new=03H
        RET

;-------------------------------------------------------

; 按键处理子程序

;-------------------------------------------------------
PROC_KEY:
        MOV   A,KEY_NEW
        JNB   ACC.0,PROC_K1          ;K1 键按下
        JNB   ACC.1,PROC_K2          ;K2 键按下
     RET
PROC_K1:                             ;按键 K1 处理程序
        CLR   RELAY                 ;继电器吸合
        RET

PROC_K2:                            ;按键 K2 处理程序
        SETB   RELAY                 ;继电器释放
        RET

;-------------------------------------------------------

; 延时子程序(10MS)

;-------------------------------------------------------
DELAY:
        MOV   R6,#10
DEL1:
        MOV   R7,#185
```

```
DEL2:
        NOP
        NOP
        NOP
        DJNZ   R7,DEL2
        DJNZ   R6,DEL1
        RET

;-------------------------------------------------------

        END       ;结束

;-------------------------------------------------------
```

6. C 语言源程序

(Example_C51\EX3_RELAY)

```
#include <reg52.h>

char code reserve[3]_at_ 0x3b; //保留 0x3b 开始的 3 个字节

sbit K1 = P1 ^ 4;
sbit K2 = P1 ^ 5;
sbit relay = P3 ^ 6;

unsigned char key_new, key_old;

/-------------------------------------------------------

延时函数

 -------------------------------------------------------/
void delayms(unsigned int ms)
{
  unsigned char k;
  while (ms--)
  {
    for (k = 0; k < 114; k++);
  }
}

/-------------------------------------------------------
扫描健盘函数
 -------------------------------------------------------/
unsigned char scan_key()
{
  key_new = 0x00;
  key_new |= K2;
  key_new <<= 1;                  //左移 1 位
  key_new |= K1;
  return key_new;               //无键按下 key_new=0x03。
}

/-------------------------------------------------------
主函数
 -------------------------------------------------------/
void main(void)
{
  P0 = 0xff;                 //初始化端口
```

```
P2 = 0xff;
P1 = 0xf0;                    //置 P1 高四位为输入
key_old = 0x03;               //初始键比较值
relay = 1;                    //继电器释放

while (1)
{
  scan_key();
  if (key_new != key_old)
  {
    delayms(10);              //延时去抖动
    scan_key();               //再次判健是否按下
    if (key_new != key_old)
    {
      key_old = key_new;          //保存按键状态
      if ((key_new &0x01) == 0)   //K1 键按下
        relay = 0;                //继电器吸合

          if ((key_new &0x02) == 0)     //K2 键按下

            relay = 1;                  //继电器释放

        }

      }

    }

  }
```

实验四　蜂鸣器控制

蜂鸣器是一种一体化结构的电子讯响器，广泛应用于电子产品中作发声报警。

蜂鸣器有两类：一类是压电式，一类是电磁式。

因为 ME850 上使用是电磁式蜂鸣器，所以只对电磁式蜂鸣器进行相应的介绍。

电磁式蜂鸣器有两种类型：有源蜂鸣器和无源蜂鸣器。有源蜂鸣器内部带振荡源，无源蜂鸣器内部不带振荡源。**这里所说的"源"不是指"电源"，而是指"振荡源"。**

有源蜂鸣器和无源蜂鸣器的主要差别是对输入信号的要求不一样，有源蜂鸣器工作的理想信号是直流电，无源蜂鸣器工作的理想信号是方波。无源蜂鸣器接直流电是不会工作的。

1. 实验任务

蜂鸣器响 300ms，P0 端口的 D00 和 D07 点亮。

蜂鸣器停 300ms，P0 端口的 D00 和 D07 熄灭。

蜂鸣器在不停地一响一停，发光二极管也在不停地一亮一灭，间隔时间为 300ms，形成声光报警效果。

2. 实验线路

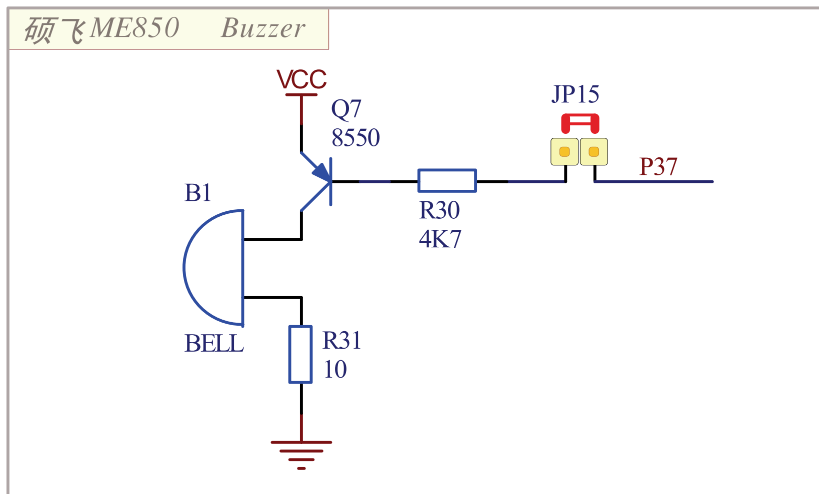


图 5.7 蜂鸣器实验电路

由于蜂鸣器工作时电流比较大，单片机的 I/O 端口输出电流比较小，无法直接进行驱动，需要加一个 PNP 型三极管 8550 进行隔离放大。

单片机的 P3.7 引脚通过限流电阻 R30 与三极管 Q7 的基极相接，蜂鸣器 B1 和电阻 R31 串接在三极管 Q7 的集电极回路中。

三极管 Q7 的基极经限流电阻 R30 后由单片机的 P3.7 引脚控制，当 P3.7 输出高电平时，三极管 Q7 截止，没有电流流过蜂鸣器内部线圈，蜂鸣器不发声；当 P3.7 输出低电平时，三极管 Q7 导通，有电流流过蜂鸣器内部线圈，蜂鸣器发出声。所以让 P3.7 引脚不断地输出方波，三极管 Q7 就会不断地导通和截止，使无源蜂鸣器发出声音。

因此，可以通过程序控制 P3.7 引脚电平来使蜂鸣器发音与关闭。

程序中改变单片机 P3.7 引脚的输出频率，就可以调整蜂鸣器的音调，可产生各种不同音调的声音。改变 P3.7 引脚的输出电平的占空比，则可以控制蜂鸣器的声音大小。

3. 实验步骤

短接 JP15 短接子，使蜂鸣器接口电路使能。

将 JP9 的 VCC - VCC3 短接子用短接帽短接，VCC 向发光二极管模块供电。

4. 程序流程图

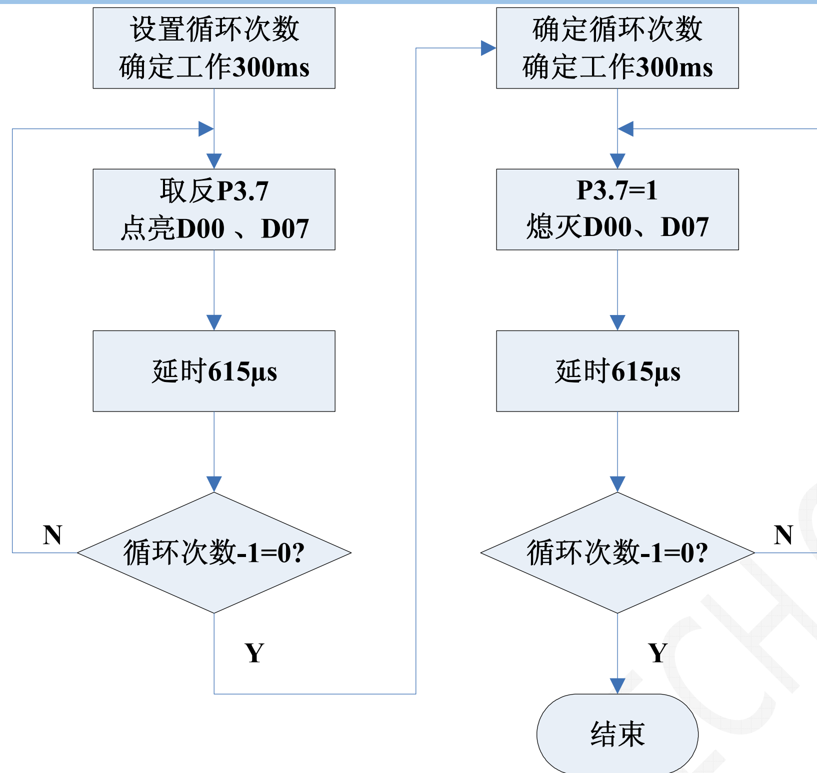


图 5.8　蜂鸣器实验流程图

5.汇编源程序

（Example_A51\EX4_BELL）

```
BEEP   BIT   P3.7

            ORG 0000H
            AJMP   MAIN
            ORG   0050H

MAIN:
        MOV   SP,#60H            ;初始化
        MOV   P0,#0FFH
        MOV   P2,#0FFH
LOOP1:
        MOV   R2,#10             ;响 300ms
LOOP2:
        MOV   R3,#49
LOOP3:
        CPL BEEP                 ;输出频率 800Hz
        MOV   P0,#7EH            ;点亮 D00 和 D07
        ACALL DELAY              ;延时 615us
        DJNZ R3,LOOP3
        DJNZ R2,LOOP2

        MOV   R2,#10             ;关 300ms
LOOP4:
        MOV   R3,#49
LOOP5:
        SETB   BEEP              ;关闭蜂鸣器
        MOV   P0,#0FFH           ;关闭显示
```

```
        ACALL DELAY         ;延时 615us
        DJNZ R3,LOOP5
        DJNZ R2,LOOP4

        SJMP LOOP1

;-----------------------------------------------------------
; 延时 615us
;-----------------------------------------------------------
DELAY:
        MOV R7,#189
DEL:
        NOP
        DJNZ R7,DEL
        RET

        END                 ;结束

;-----------------------------------------------------------
;信号产生的方法
;800Hz 信号周期为 1230us
;615-49-10=301350us=301ms
;-----------------------------------------------------------
```

6. C 语言源程序

（Example_C51\EX4_BELL）

```
#include <reg52.h>
#include <intrins.h>

sbit BEEP = P3 ^ 7;
char code reserve[3]_at_ 0x3b; //保留 0x3b 开始的 3 个字节

/*---------------------------------------------------------
310us 延时函数
晶振：11.0592MHz
 ---------------------------------------------------------*/
void delay(void)
{
  unsigned char i;
  for (i = 143; i > 0; i--)      ;
}

/*---------------------------------------------------------
主函数
 ---------------------------------------------------------*/
void main(void)
{
  unsigned int j;
  P0 = 0xff; //端口初始化
  P1 = 0xff;
  P2 = 0xff;

  while (1)
  {
    for (j = 490; j > 0; j--)
    //响 300ms
    {
```

```
        BEEP = ~BEEP; //输出频率 800Hz
        P0 = 0x7E; //点亮 D00 和 D07
        delay(); //延时 310us
        delay(); //延时 310us
      }

      for (j = 490; j > 0; j--)
      //关 300ms
      {
        BEEP = 1; //关闭蜂鸣器
        P0 = 0xff; //关闭显示
        delay(); //延时 310us
        delay(); //延时 310us
      }
    }
}
```

实验五　数码管显示 0-7

数码管显示输出是单片机系统中最常用的一种显示输出，主要用于单片机控制中的数据输出和状态信息显示。

1. 实验任务

先将"0-7"数码管的段码值写入显示存储器中，使 8 位数码管从右至左显示 0–7 。

2. 实验线路

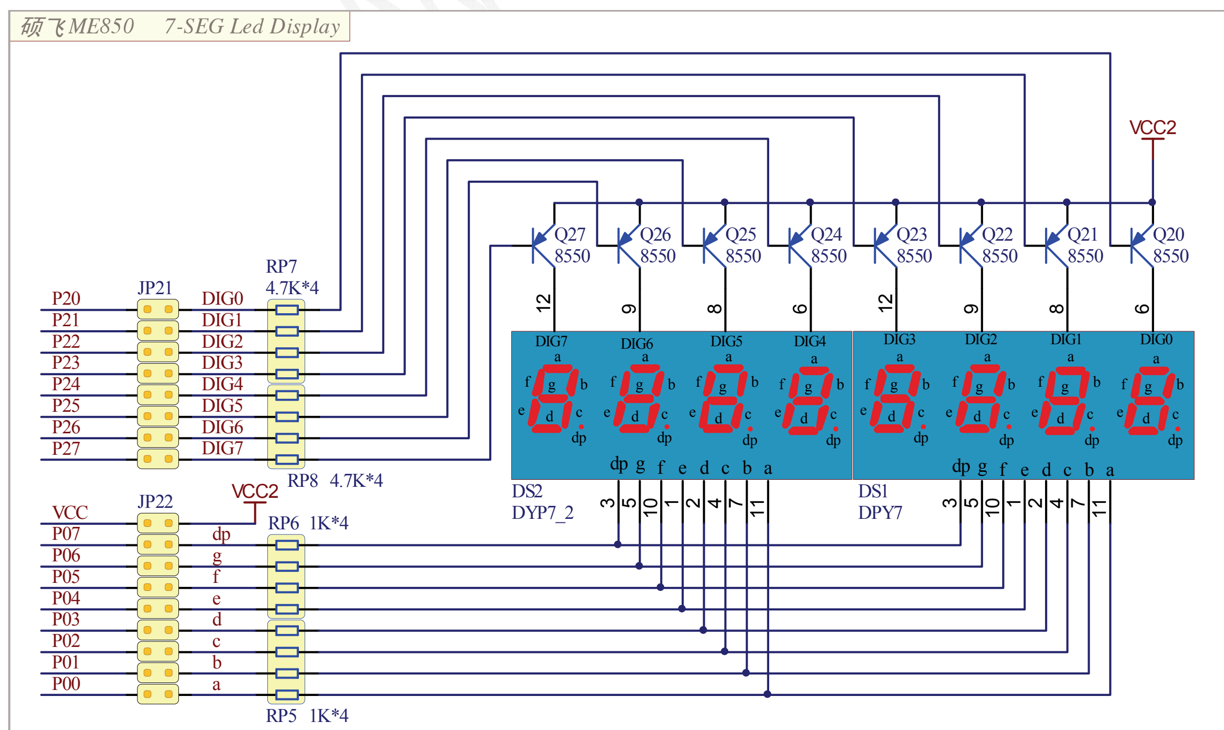


图 5.9　数码管显示电路

ME850 采用 8 位数码管动态扫描显示，可以简化硬件电路、方便软件编程和减少电源的功耗。具体的电路原理图如图 1 所示。

8 位数码管动态扫描显示,就是将 8 个数码管的 8 个相同的段线并接在一起，通过限流电阻 RP5、RP6 接到 AT89S52 的 P0 口，由 P0 口控制字段输出(低电平有效)。而各位共阳极数码管的 COM 由 AT89S52 的 P2 口(低电平有效)通过限流电阻 RP7、RP8 控制 Q20 - Q27 来实现 8 位数码管的位输出控制(高电平有效)。

2.1 数码管动态扫描原理简介

从数码管动态扫描显示电路的原理可知，对于 8 位数码管动态扫描显示需要由两组信号来控制：**一组是字段输出口输出的字形代码，用来控制显示的字形，称为段码；另一组是位输出口输出的控制信号，用来选择第几位数码管工作，称为位码。**

由于各位数码管的段线并联，段码的输出对各位数码管来说都是相同的。因此，在同一时刻如果各位数码管的位选线都处于选通状态的话，8 位数码管将显示相同的字符。若要各位数码管能够显示出与本位相应的字符，就必须采用扫描显示方式。即在某一时刻，只让某一位的位选线处于导通状态，而其它各位的位选线处于关闭状态。同时，段线上输出相应位要显示字符的字型码。这样在同一时刻，只有选通的那一位显示出字符，而其它各位则是熄灭的，如此循环下去，就可以使各位数码管显示出将要显示的字符。

虽然这些字符是在不同时刻出现的，而且同一时刻，只有一位显示，其它各位熄灭，但由于数码管具有余辉特性和人眼有视觉暂留现象，只要每位数码管显示间隔足够短，给人眼的视觉印象就会是连续稳定地显示。

数码管不同位显示的时间间隔可以通过调整延时程序的延时长短来完成。数码管显示的时间间隔也能够确定数码管显示时的亮度，若显示的时间间隔长，显示时数码管的亮度将亮些，若显示的时间间隔短，显示时数码管的亮度将暗些。若显示的时间间隔过长的话，数码管显示时将产生闪烁现象。所以，在调整显示的时间间隔时，即要考虑到显示时数码管的亮度，又要数码管显示时不产生闪烁现象。

2.2 数码管组成结构

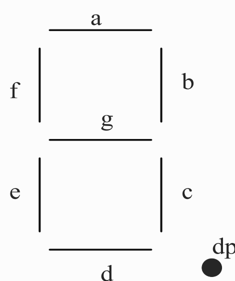


图 5.10　数码管结构图

数码管内部是由 7 个条形的发光二极管和右下方一个圆形的发光二极管组成，这样一共有 8 个段线，恰好适用于 8 位的并行系统。数码管按内部连接方式分为共阴极数码管和共阳极数码管两种。

（1）共阴极数码管

共阴极数码管是将所有发光二极管的阴极接在一起作为公共端 COM，当公共端接低电平时，某一段阳极上的电平为"1"时，该段点亮，电平为"0"时，该段熄灭。

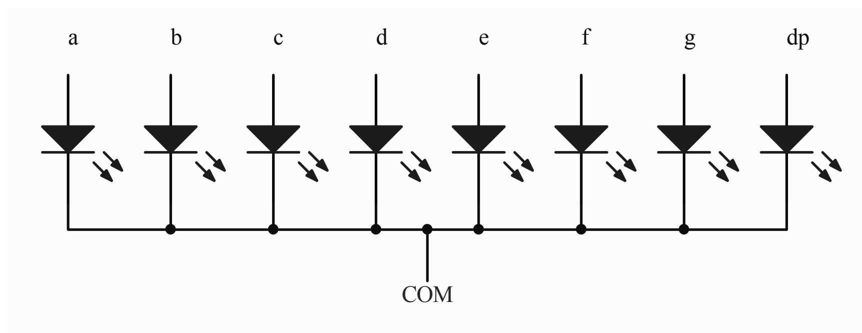


图 5.11 共阴数码管结构图

（2）共阳极数码管

共阳极数码管是将所有发光二极管的阳极接在一起作为公共端 COM，当公共端接高电平时，某一段阴极上的电平为"0"时，该段点亮，电平为"1"时，该段熄灭。

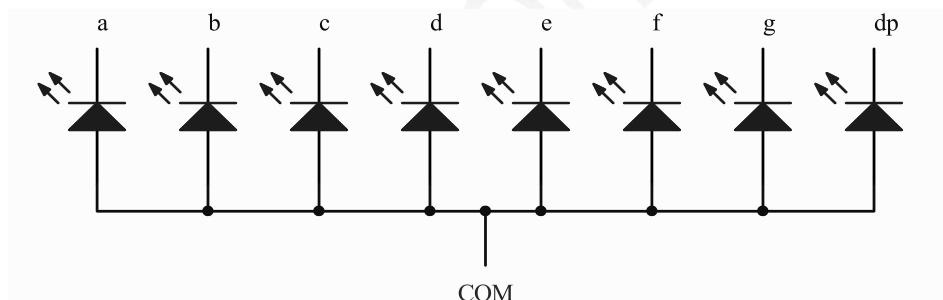


图 5.12 共阳数码管结构图

（3）共阳极数码管的字型代码表

字型	dp	g	f	e	d	c	b	a	段码
0	1	1	0	0	0	0	0	0	0C0H
1	1	1	1	1	1	0	0	1	0F9H
2	1	0	1	0	0	1	0	0	0A4H
3	1	0	1	1	0	0	0	0	0B0H
4	1	0	0	1	1	0	0	1	99H
5	1	0	0	1	0	0	1	0	92H

6	1	0	0	0	0	0	1	0	82H
7	1	1	1	1	1	0	0	0	0F8H
8	1	0	0	0	0	0	0	0	80H
9	1	0	0	1	0	0	0	0	90H
a	1	0	0	0	1	0	0	0	88H
b	1	0	0	0	0	0	1	1	83H
c	1	1	0	0	0	1	1	0	0C6H
d	1	0	1	0	0	0	0	1	0A1H
E	1	0	0	0	0	1	1	0	86H
f	1	0	0	0	1	1	1	0	8EH

举例：

如果你想让图 1 最右边的数码管显示"0"的话，首先将段码"0C0H"送达 P0 口，然后将 P2.0 清为低电平。当 P2.0 为低电平时，三极管 Q20 导通，其该位数码管的公共阳极接至＋5V，于是该位数码管就显示"0"。

```
MOV   P0,#0C0H      ;送段码到 P0 口
MOV   P2,#0FEH      ;清 P2.0 为低电平
```

3. 实验步骤

将 JP21 的 8 个短接子用短接帽短接，使数码管的位控制线与 P2 端口接通。

将 JP22 的 9 个短接子用短接帽短接，使数码管的数据线与 P0 端口接通，并使 VCC 向数码管接口电路供电。

将 JP24 的 OFF 端用短接帽短接，禁止 LCD1602 显示功能，否则数码管将不能正常显示。

4. 程序设计

数码管显示程序的编程方法

1）先准备好要显示的数据，放入相应的显示存储单元中。

2）根据要使用的数码管的具体位置来确定扫描初值和扫描方向。

3）根据使用数码管的个数来确定扫描的位数。

4）查表将要显示的数据转换为能使数码管正确显示相对应的段码。

5）分时送段码和位码，数码管开始循环显示。

5.程序流程图

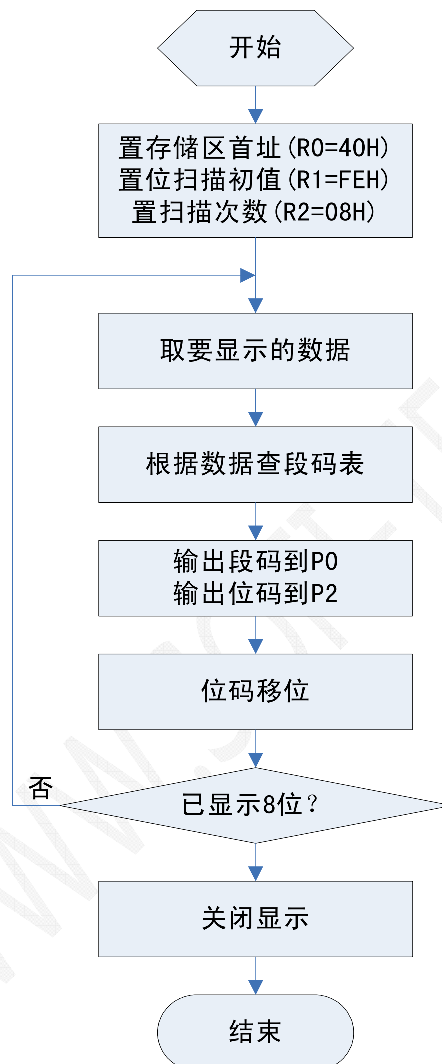


图 5.13　EX5_7SEG 流程图

6. 汇编源程序

（Example_A51\EX5_7SEG）

```
DISSTART   EQU   40H        ;显示单元首地址
           LED_DATA   EQU   P0        ;数码管数据口定义

;-----------------------------------------------------------

           ORG 0000H
           AJMP MAIN
```

```
          ORG 0050H
;-----------------------------------------------------------
; 主程序
;-----------------------------------------------------------
MAIN:
          MOV   SP,#60H
          MOV   P0,#0FFH          ;初始化
          MOV   P2,#0FFH
          MOV   R2,#08H           ;8 组数据
MOV   R0,#DISSTART              ;显示存储单元首地址
          MOV   R1,#00H
MAIN1:
          MOV   A,R1
          MOV   @R0,A             ;将 0-7 分别存入显示存储单元
          INC   R1
          INC   R0
          DJNZ   R2,MAIN1         ;
LOOP:
          ACALL   PLAY            ;循环显示
          SJMP    LOOP
;-----------------------------------------------------------
; 显示子程序
;-----------------------------------------------------------
PLAY:
         MOV R0,#DISSTART          ;获得显示单元首地址
         MOV R1,#0FEH              ;位码初始值
         MOV R2,#08H               ;有 8 位数码管显示
DISP1:
         MOV A,@R0                 ;取要显示的数据
         MOV DPTR,#TAB_NU          ;置段码表首址
         MOVC A,@A+DPTR            ;根据数据查段码表
         MOV LED_DATA,A            ;段码输出
         MOV P2,R1                 ;位码输出
         MOV A,R1                  ;准备下一次显示的位码
         RL A
         MOV R1,A                  ;保存位码
         INC R0                    ;取下一个显存单元地址
         ACALL DELAY               ;调用延时
         DJNZ R2,DISP1             ;8 位数码管是否显示完
         MOV   P2,#0FFH            ;关闭显示
         RET                       ;显示完成，返回
;----------------------------------------------------------
;1MS 延时子程序
;----------------------------------------------------------
DELAY:
         MOV   R6,#5
DEL1:
         MOV   R7,#93
```

```
DEL2:
        DJNZ   R7,DEL2           ;第一层循环
        DJNZ   R6,DEL1           ;第二层循环
        RET

;---------------------------------------------------------

; 段码表

;---------------------------------------------------------
TAB_NU:
    DB 0C0H,0F9H,0A4H,0B0H,099H,092H,082H,0F8H,080H
    DB 090H,088H,083H,0C6H,0A1H,086H,08EH,0FFH

;---------------------------------------------------------

        END                    ;结束

;---------------------------------------------------------
```

7. C 语言源程序

（Example_C51\EX5_7SEG）

```
#include <reg52.h>
#include <intrins.h>

char code reserve[3]_at_ 0x3b; //保留 0x3b 开始的 3 个字节

unsigned char code display[] =
{
  0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8, 0x80, 0x90
};

/---------------------------------------------------------
 延时子程序
 ---------------------------------------------------------/
void delayms( unsigned int ms )
{
  unsigned char k;
  while (ms--)
  {
    for (k = 0; k < 114; k++)
      ;
  }
}

/---------------------------------------------------------
 显示函数
 ---------------------------------------------------------/
void main(void)
{
  unsigned char k, shift;

  P0 = 0xff;                     //端口初始化
  P2 = 0xff;
```

```
    while (1)
    {
      shift = 0xfe;              //位扫描初值
      P2 = 0xff;                 //关闭显示
      for (k = 0; k < 8; k++)
      {
        P0 = display[k];         //送段码
        P2 = shift;              //送位码
        shift = _crol_(shift, 1);  //左移一位，修改位码
        delayms(1);              //延时 1ms
      }
    }
}

/--------------------------------------------------------/
```

实验六　独立按键识别

按键（轻触开关）是一种广泛应用于各种电子设备的元件，比如我们最常用的电视机面板控制按钮，遥控器按钮。其实就是一个常开的开关，按下后两个触点接触形成通路状态，松开时形成开路状态。

1. 实验任务

当有键按下，对应的 LED 灯亮。

K1 - K8 对应 P0 端口的 LED　D00 - D07

K1 键按下后，D00 亮。

……

K8 键按下后，D07 亮

在确认有按键按下时，蜂鸣器会响一声。

2. 实验电路

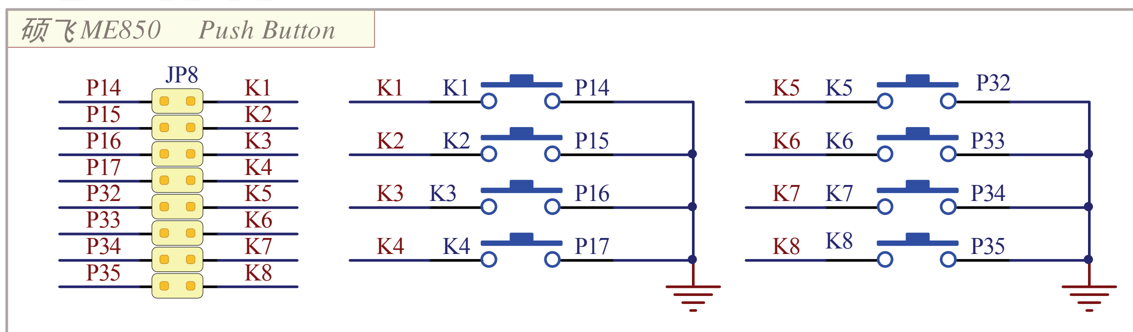


图 5.14　ME850 独立按键

ME850 上共有 8 个独立按键 K1 - K8 ，分别通过短接跳线组 JP8 连接到 P14-P17、P32 - P35 端口

LED 显示电路已经在前面介绍，见图 5.1

3．实验步骤

将 JP8 的 8 个短接子全部用短接帽短接，使独立按键与相应的端口接通；

将 JP9 的 VCC-VCC3 短接子用短接帽短接，使 VCC 向发光二极管 D00-D07 供电；

将 JP15 短接子用短接帽短接，使蜂鸣器接口电路工作使能；

如果 PS2 接口插了键盘，请拔下键盘插头或者取下 JP12 上的短接帽。

4. 程序流程图

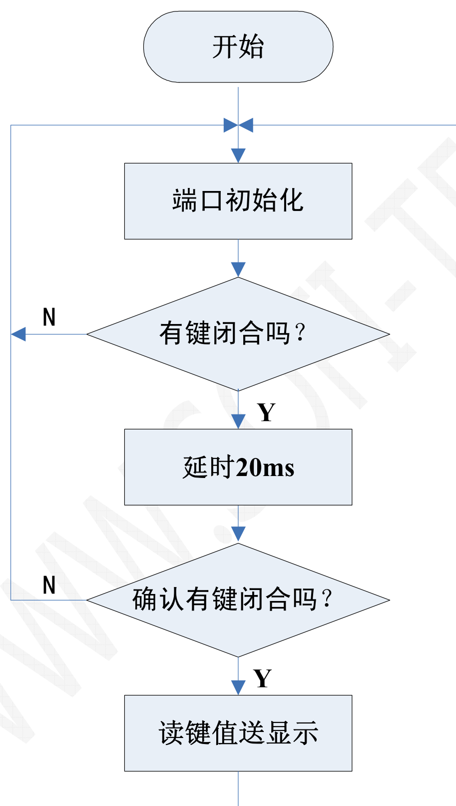


图 5.16　EX6_KEY 流程图

5.汇编源程序

（Example_A51\EX6_KEY）

```
        BEEP   BIT    P3.7
;-------------------------------------------------------------

        ORG   0000H
        AJMP   MAIN
        ORG   0050H

;-------------------------------------------------------------
MAIN:
```

```
        MOV   SP,#60H
        MOV   P2,#0FFH
LOOP0:
        MOV   P0,#0FFH          ;关闭所有 LED 显示
        MOV   P1,#0FFH          ;为输入状态
        MOV   P3,#0FFH

        ACALL   SCANKEY           ;第一次判键
        CJNE   A,#0FFH,LOOP1
        AJMP   LOOP0
LOOP1:
        MOV   R5,#2              ;延时 20ms
        ACALL   DELAY
        ACALL   SCANKEY           ;第二次判键
        CJNE   A,#0FFH,LOOP2
        AJMP   LOOP0

LOOP2:
        MOV   P0,A              ;键值送显示
         ACALL   BEEP_BL
        AJMP   LOOP0

;-------------------------------------------------------

; 键盘扫描子程序

;-------------------------------------------------------
SCANKEY:
        MOV   P1,#0FFH        ;为输入状态
        MOV   P3,#0FFH

        MOV   R0,P1           ;读 P1 端口
        MOV   R1,P3           ;读 P3 端口

        MOV   A,R0
        ANL   A,#0F0H         ;保留高四位
        SWAP   A              ;低四位与高四位数据互换
        MOV   R0,A            ;低四位为有效位(K1-K4)

        MOV   A,R1
        RL   A                ;左移两次
        RL   A
        ANL   A,#0F0H         ;高四位为有效位(K5-K8)

        ORL   A,R0            ;重新组合，键值保存在 A
        RET

;--------------------------------------------------------
;
;蜂鸣器响一声子程序
;
;--------------------------------------------------------
BEEP_BL:
         MOV   R6,#200
BL1:
         ACALL   BL2
         CPL   BEEP              ;蜂鸣器取反产生驱动脉冲
         DJNZ   R6,BL1
         SETB   BEEP              ;关闭蜂鸣器
```

```
          MOV   R5,#15              ;延时 150ms,防止键连击
          ACALL   DELAY
          RET
BL2:
          MOV   R7,#220
BL3:
          NOP
          DJNZ   R7,BL3
          RET

;--------------------------------------------------------

; 延时子程序

;--------------------------------------------------------
DELAY:                              ;延时 R5×10MS
          MOV   R6,#50
DEL1:
          MOV   R7,#93
DEL2:
          DJNZ   R7,DEL2
          DJNZ   R6,DEL1
          DJNZ   R5,DELAY
          RET

;-------------------------------------------------------

          END                    ;结束

;-------------------------------------------------------
```

6. C 语言源程序

（Example_C51\EX6_KEY）

```
#include <reg52.h>

char code reserve[3]_at_ 0x3b; //保留 0x3b 开始的 3 个字节

sbit BEEP = P3 ^ 7;

/*******************************************************

延时子函数

 *******************************************************/
void delayms(unsigned int ms)
{
  unsigned int t;
  while (ms--)
  {
    for (t = 0; t < 114; t++)
      ;
  }
}

/*******************************************************

x*0.14MS  延时子函数

 *******************************************************/
```

```
void delayus(unsigned char x)
{
  unsigned char i;
  while (x--)
  {
    for (i = 0; i < 14; i++)
    {
      ;
    }
  }
}

/*****************************************************

蜂鸣器驱动子函数

 *****************************************************/
void beep()
{
  unsigned char i;
  for (i = 0; i < 200; i++)
  {
    delayus(6);
    BEEP = !BEEP; //BEEP 取反
  }
  BEEP = 1; //关闭蜂鸣器
  delayms(150); //延时
}

/*****************************************************

键盘扫描子函数

 *****************************************************/
unsigned char scankey()
{
  unsigned char keynum, keynum1, keynum2;
  P1 = 0xff;                    //为输入状态
  P3 = 0xff;                    //为输入状态

  keynum1 = P1;            //读 P1 端口
  keynum2 = P3;             //读 P3 端口

  keynum1 = (keynum1 &0xf0) >> 4;              //低四位为有效位(K1-K4)
  keynum2 = (keynum2 &0x3c) << 2;              //高四位为有效位(K5-K8)

  keynum = keynum1 | keynum2;                   //组合

  return (keynum);
}

/*****************************************************

主函数

 *****************************************************/
void   main()
{
  unsigned char key;
  P0 = 0xff;                               //关闭 LED 显示
  P2 = 0xff;
```

```
  while (1)
  {
    key = scankey();            //第一次判断
    if (key != 0xff)
    {
      delayms(20);              //延时 20ms
      key = scankey();          //第二次判断
      if (key != 0xff)
      {
        P0 = key;               //键值送显示
        beep();
      }
    }
    P0 = 0xff;                  //关闭 LED 显示
  }
}

/*************************************************************/
```


实验七　外部中断

1. 实验任务

利用单片机的外部中断功能进行计数，然后将计数值输出到数码管上显示。

K5 键 - 计数值加 1（外部中断 0）

K6 键 - 计数值减 1（外部中断 1）

3 位数码管显示，最大计数值 255。

2. 实验线路

见前面章节

3. 实验步骤

将 JP21 的 8 个短接子全部用短接帽短接，使 DG0～DG7 与 P2 端口接通。

将 JP22 的 9 个短接子全部用短接帽短接，使 A～DP 与 P0 端口接通，VCC 向数码管模块供电。

将 JP8 的 8 个短接子全部用短接帽短接，使独立按键与相应的端口接通。

将 1602LCD 的使能跳线 JP24 短接在 OFF 端以禁用 1602LCD，避免 1602LCD 干扰数码管显示

4. 程序流程图

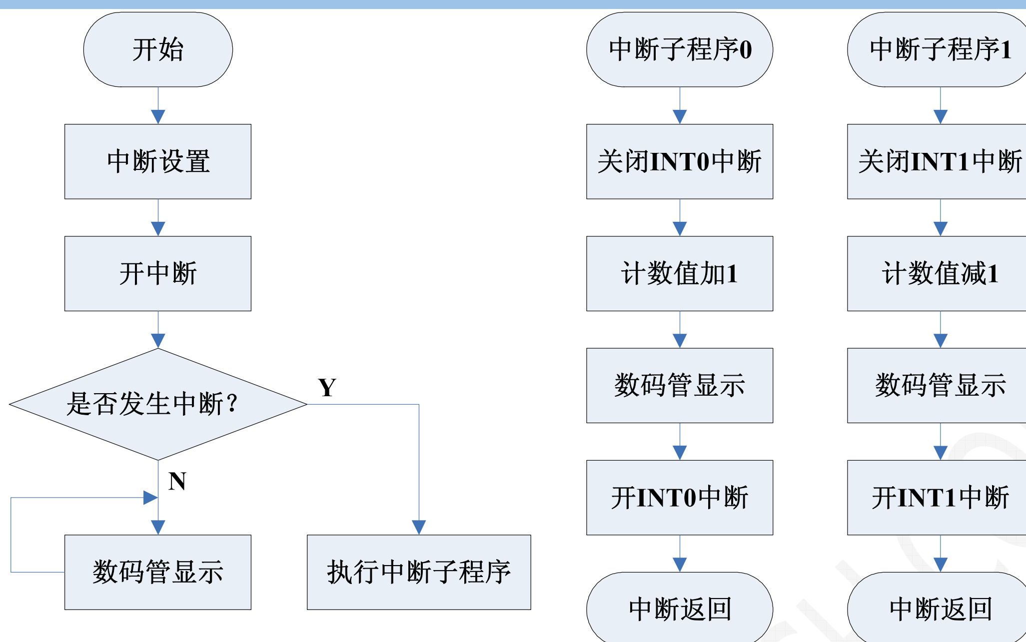


图 5.17　EX7_KEY_INT 流程图

5. 汇编源程序

见光盘 Example_A51\EX7_KEY_INT

6. C 语言源程序

见光盘 Example_C51\EX7_KEY_INT

实验八　矩阵键盘识别

1. 实验任务

1 位数码管显示矩阵键盘的按下键的键值。

开机时，数码管初始显示"-"，当键按下时，数码管显示按下键的键值，蜂鸣器响一声。

2. 实验电路

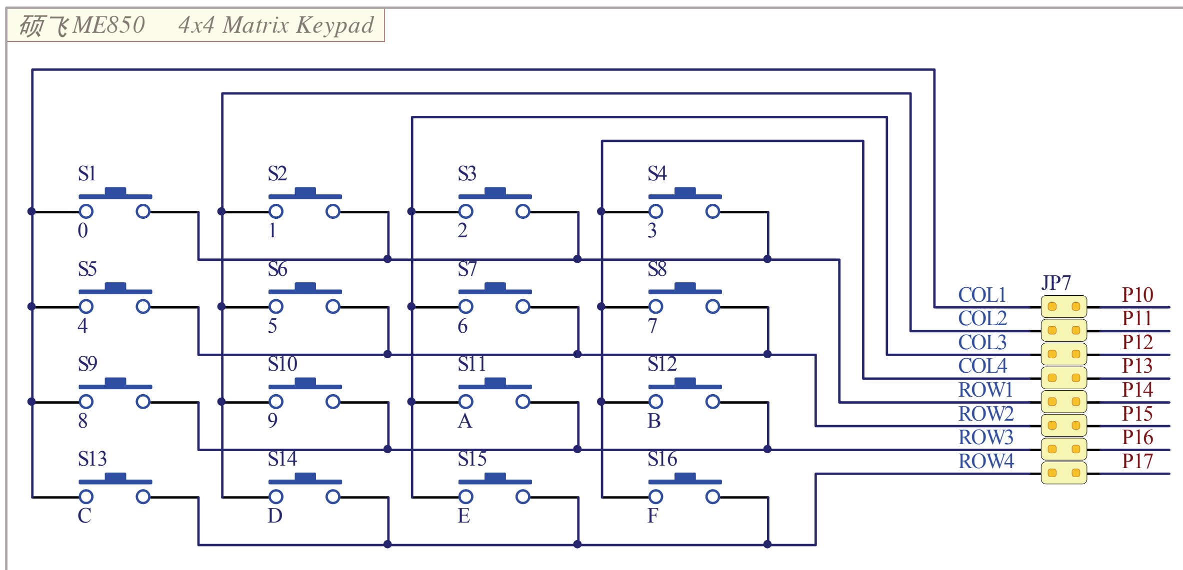


图 5.18　矩阵键盘电路

数码管部分电路请回顾前面有关章节！

3. 实验步骤

将 JP21 的 8 个短接子全部用短接帽短接，使 DG0～DG7 与 P2 端口接通。

将 JP22 的 9 个短接子全部用短接帽短接，使 A～DP 与 P0 端口接通，VCC 向数码管模块供电。

将 JP7 的 8 个短接子全部用短接帽短接，使矩阵按键与 P1 端口接通。

将 JP6（步进电机）短接子上的短接帽全部取掉，否则矩阵键盘将不能正常工作。

将 JP24 的 OFF 端短接子用短接帽短接，禁止 LCD1602 显示功能，否则数码管将不能正常显示。

4. 程序流程图

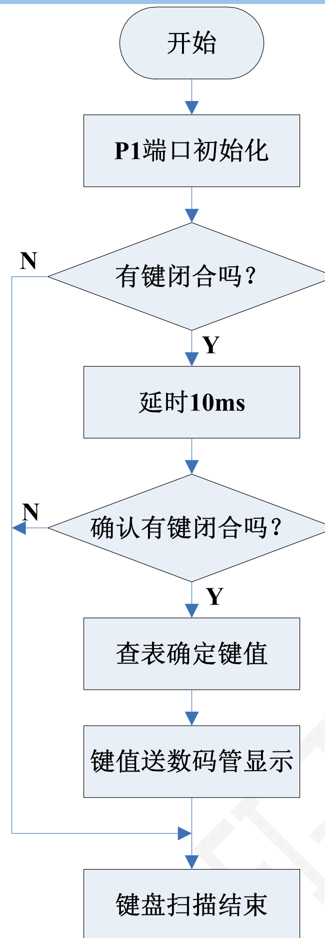


图 5.19　EX8_KEY_4X4 流程图

5. 汇编源程序

（光盘： Example_A51\ EX8_KEY_4X4 ）

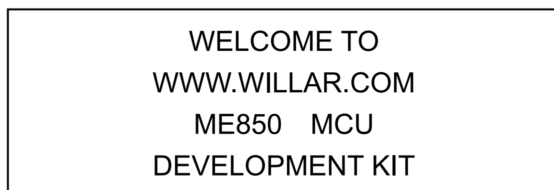
6. C 语言源程序

（光盘： Example_C51\ EX8_KEY_4X4 ）

实验九　1602 LCD 显示

1. 实验任务

在 1602LCD 上逐字移出显示两组字符串信息，如此循环。



2. 实验电路

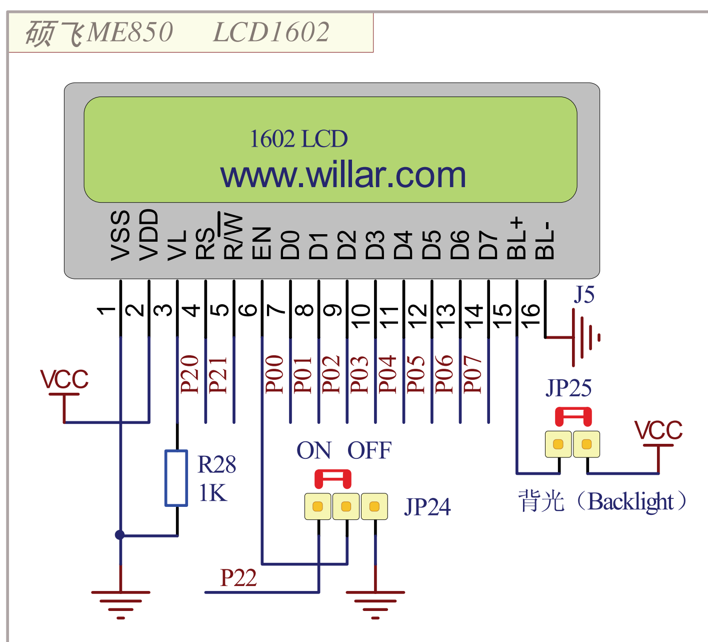


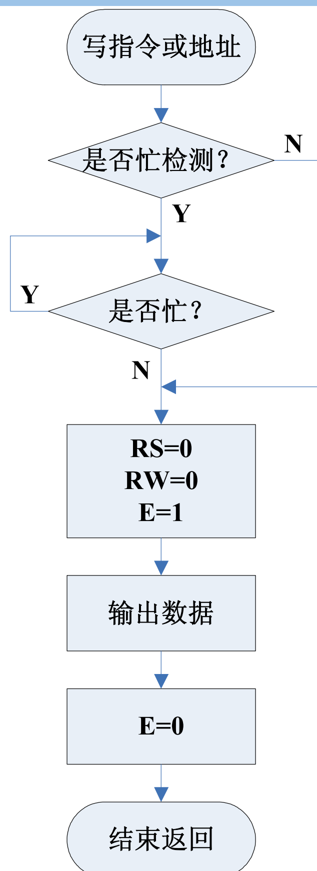
图 5.20　LCD1602 接口原理图

3. 实验步骤

将 JP24 的 ON 端 短接，使 LCD1602 使能。

将 JP25 背光短接子短接，使 LCD1602 的背光灯点亮。

4. 程序流程图



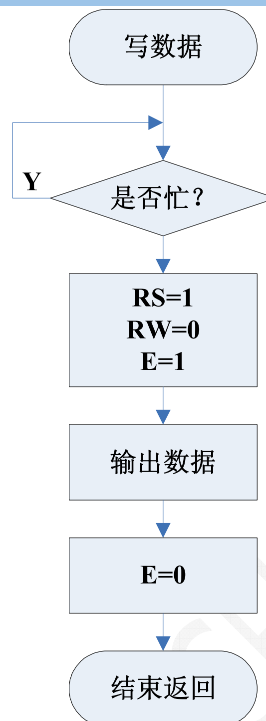


图 5.21 写指令或地址流程图　　图 5.22 写数据流程图

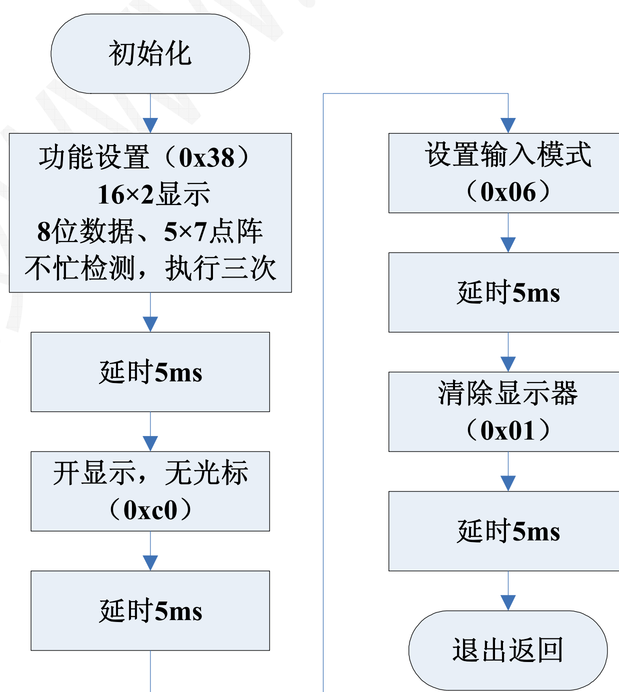


图 5.23 初始化流程图

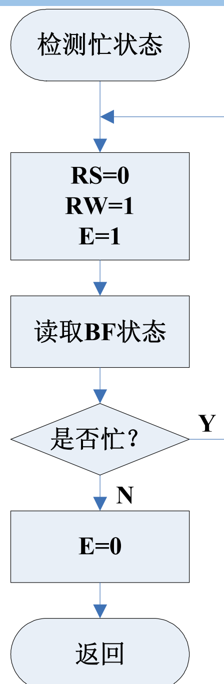


图 5.24　忙检测流程图

5. 汇编源程序

（光盘： Example_A51\EX9_LCD1602）

6. C 语言源程序

（光盘： Example_C51\EX9_LCD1602）

实验十 12864 LCD 显示

1. 实验任务

在 12864LCD 上逐字移出显示四组字符串信息和 2 张图片信息，如此循环。

硕飞科技伟纳电子
WWW.WILLAR.COM
ME850_开发实验仪
TEL:077584867757

2. 实验电路

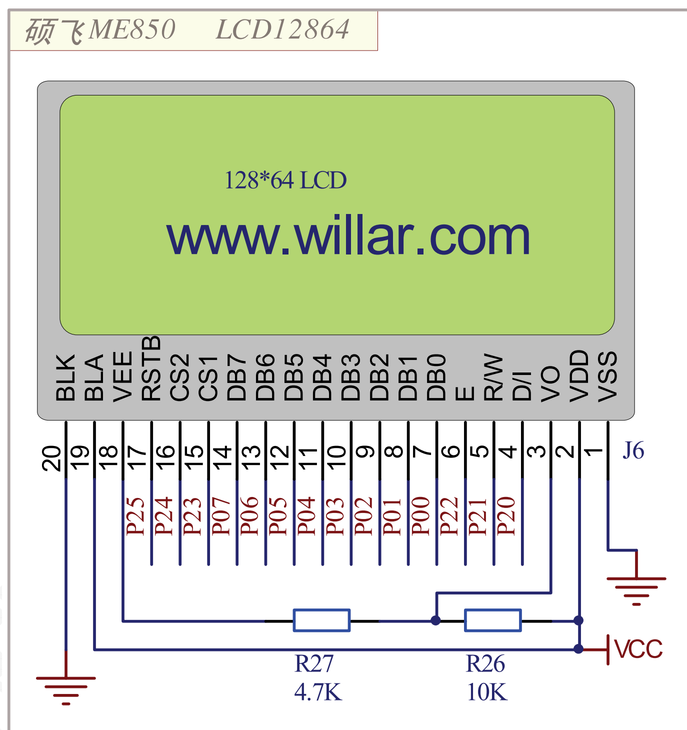


图 5.25 LCD12864 接口原理图

3. 实验步骤

将 JP25 背光短接子的短接帽取掉，使 LCD1602 的背光灯熄灭。

将 LCD12864 液晶插到 J6 插座上。

将 JP24 的 OFF 端短接，使 LCD1602 禁止。

4. 程序流程图

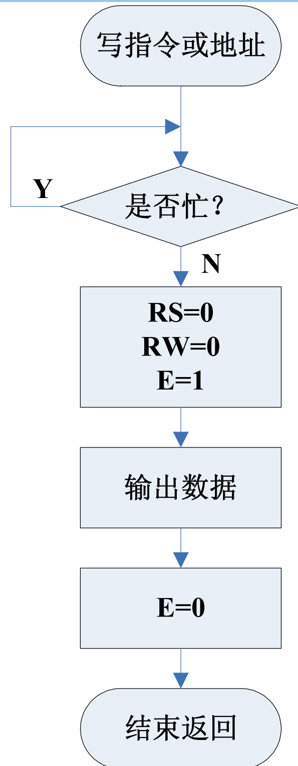


图 5.26 写指令或地址流程图

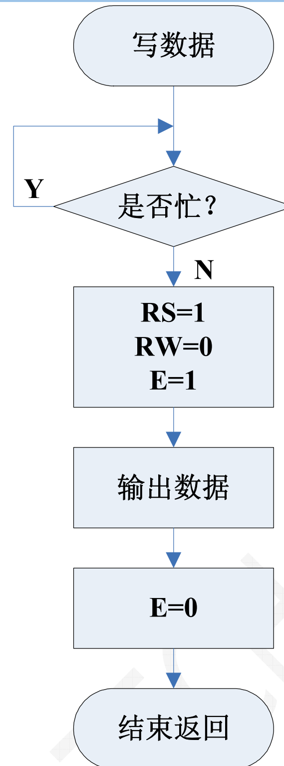


图 5.27 写数据流程图

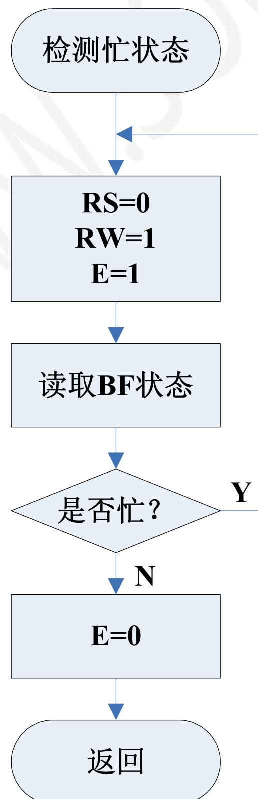


图 5.28 忙检测流程图

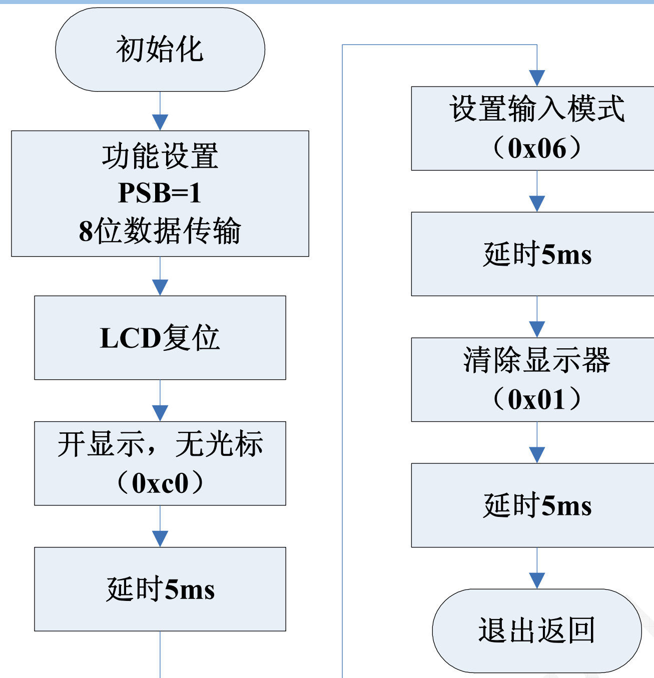


图 5.29 LCD 初始化流程图

5. 汇编源程序

（见光盘： Example_A51\EX10_LCD12864）

6. C 语言源程序

（见光盘： Example_C51\EX10_LCD12864）

实验十一 16x16 LED 点阵显示

1. 实验任务

16X16 点阵逐字显示"硕飞科技伟纳电子"，如此循环。

2. 实验电路

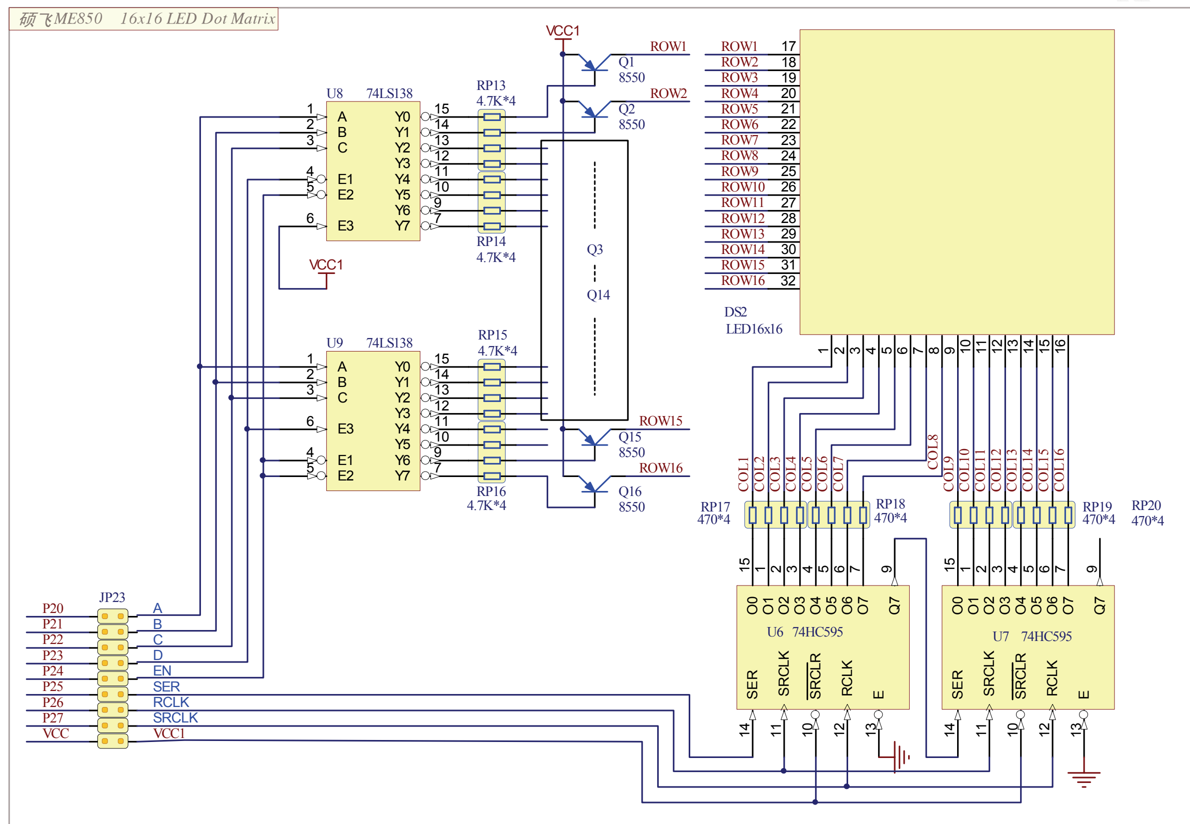


图 5.29 16x16LED 点阵原理图

3. 实验步骤

将 JP23 的 9 个短接子全部用短接帽短接，使点阵接口电路控制端与 P2 端口接通，VCC 向点阵模块供电。

4. 程序流程图

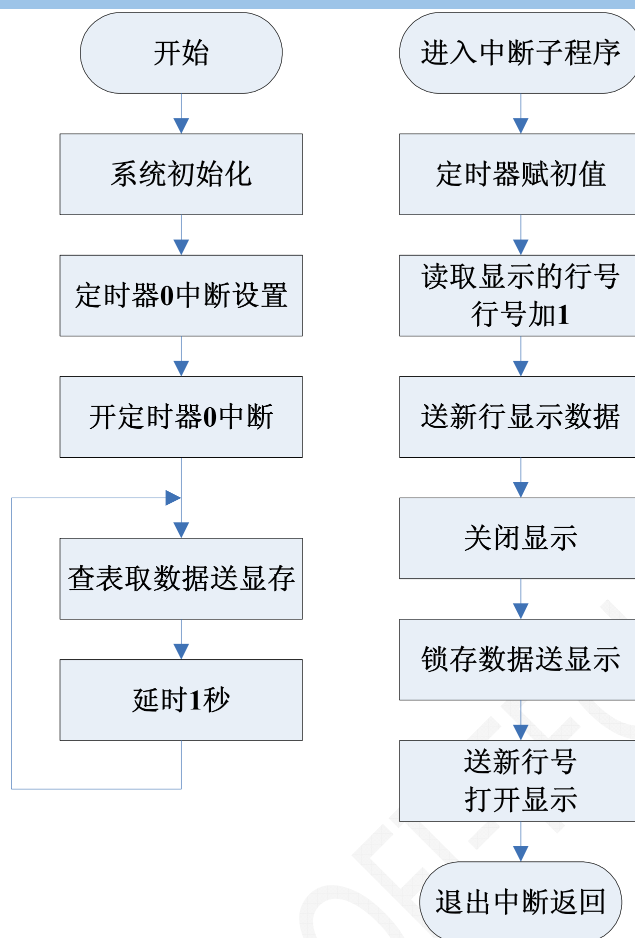


图 5.30　EX11_LED16X16 流程图

5. 汇编源程序

（见光盘： Example_A51\EX11_LED16X16）

6. C 语言源程序

（见光盘： Example_C51\EX11_LED16X16）

实验十二　RS232 串口通信

1. 实验任务

先通过串口向计算机发送中英文字符串和字符。

英文字符串： welcome to www.willar.com

中文字符串： 硕飞科技 - 伟纳电子

然后从机等待接收主机发送来的数据，当从机接收到主机发送来的数据后，将此数据再发送回主机。

2. 实验线路

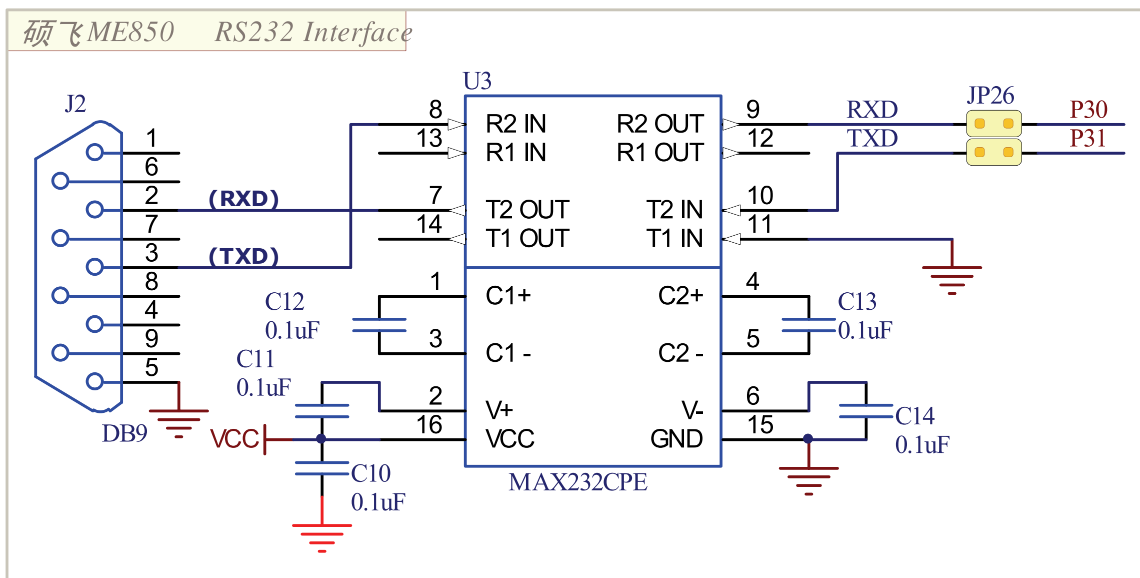


图 5.31　RS232 串口通信电路

3. 实验步骤

短接 JP26 短接子，使芯片的串行端口（RXD-P3.0、TXD-P3.1）与 RS232 接口芯片 MAX232C 连接。将 JP19 和 JP20 短接子上的短接帽全部取掉。

上位机使用伟纳编写的"串口 TT"（com TT）串口调试程序。

1）"串口 TT"参数设定：

端口号：COM1 (实际使用的端口号)　波特率：9600

数据位：8　校验位：None　停止位： 1

2）将接收信息框（左上信息框）显示模式均设置为文本模式。

3）将发送信息框（左下信息框）显示模式均设置为文本模式。

"串口 TT"软件在产品光盘"TOOL"目录中



图 5.32 串口 TT 设置

4. 程序流程图

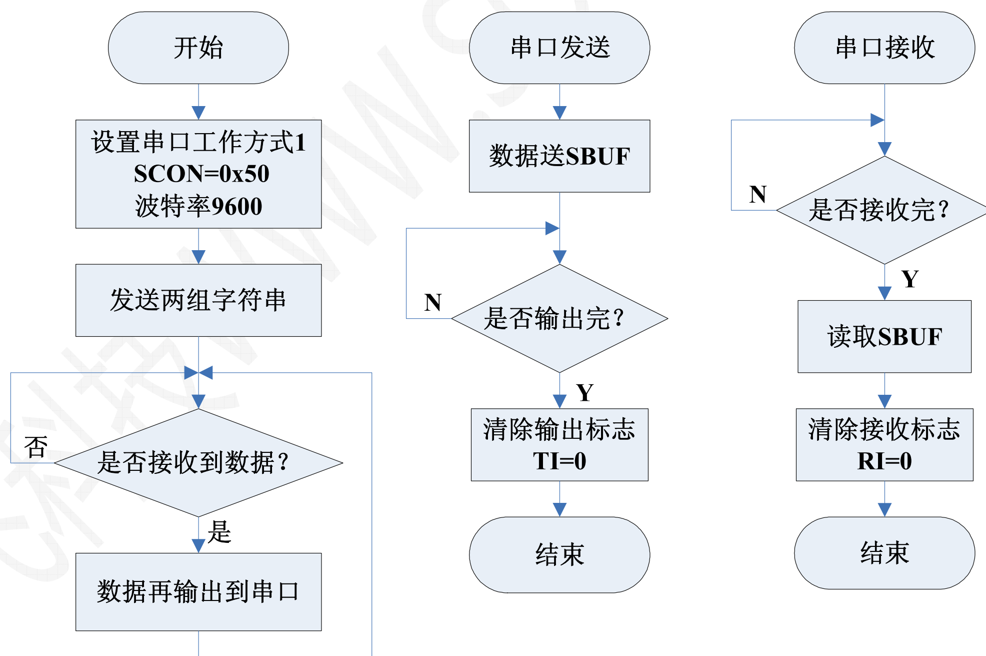


图 5.33 EX12_UART 流程图

5. 汇编源程序

（见光盘： Example_A51\EX12_UART）

6. C 语言源程序

（见光盘： Example_C51\EX12_UART）

实验十三　74HC164 串转并实验

1. 程序功能

利用 TTL 芯片 74HC164 进行串口转并口实验。

采用串口方式 0 传输数据，使 74HC164 的输出端所连接的 8 个发光二极管 DL1 - DL8 从右至左以流水方式显示。

2. 实验线路

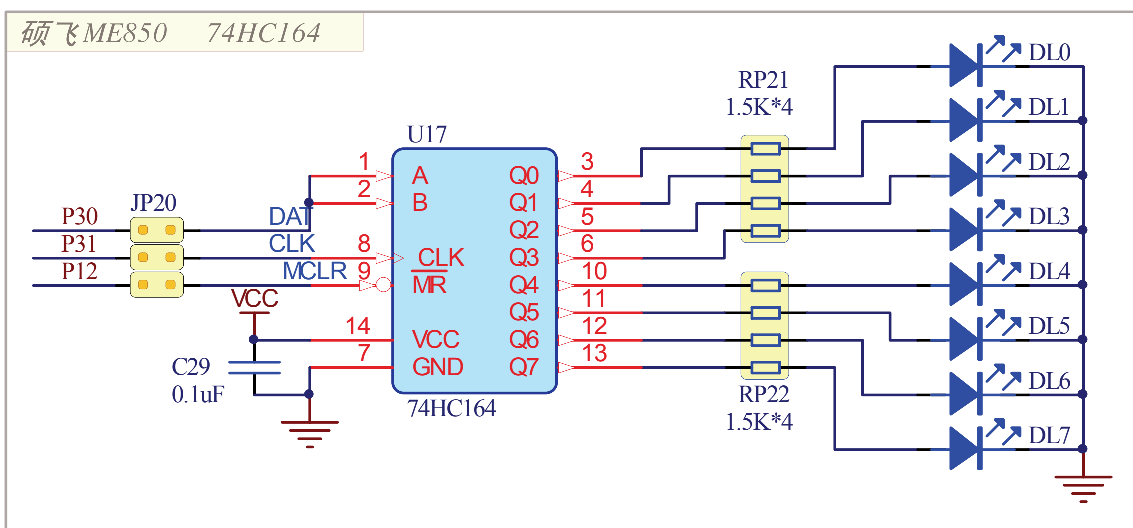


图 5.34　74HC164 原理图

3. 实验步骤

短接 JP20 短接子，使 74HC164 的相应引脚与串行端口（RXD-P3.0、TXD-P3.1）连接；

将 JP6（步进电机）、JP26（MAX232C）、JP19（74HC165）短接子上的短接帽全部取掉。

4. 程序流程图

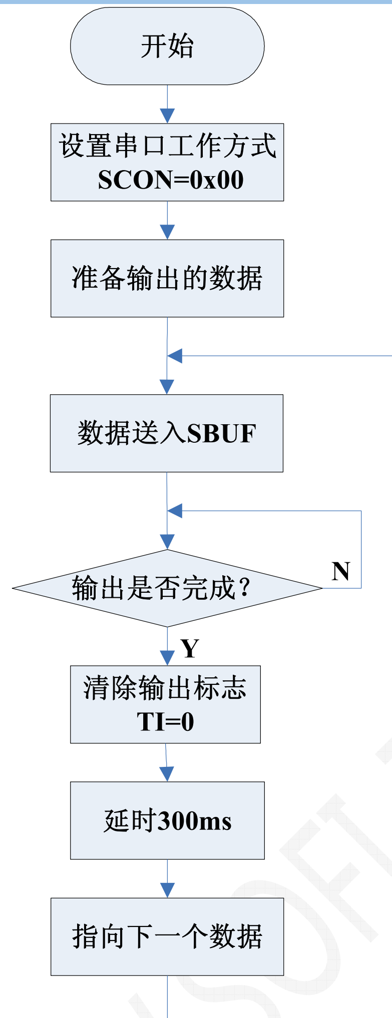


图 5.35　EX13_HC164 流程图

5. 汇编源程序

（见光盘： Example_A51\EX13_HC164）

6. C 语言源程序

（见光盘： Example_C51\EX13_HC164）

实验十四 74HC165 并转串

1. 程序功能

利用 TTL 芯片 74HC165 进行并口转串口实验。

采用串口方式 0 接收数据，读取 74HC165 并行输入口的 8 个拨指开关状态，每种状态读取 2 次，保存第一次读取值并输出到 P2 端口 D20 - D27 显示，第二次读取的值与第一次读取值进行比较，如果两次读取值相同输出到 P0 端口 D00 - D07 显示，否则关闭 P0 端口 D00 - D07 显示。

2. 实验线路

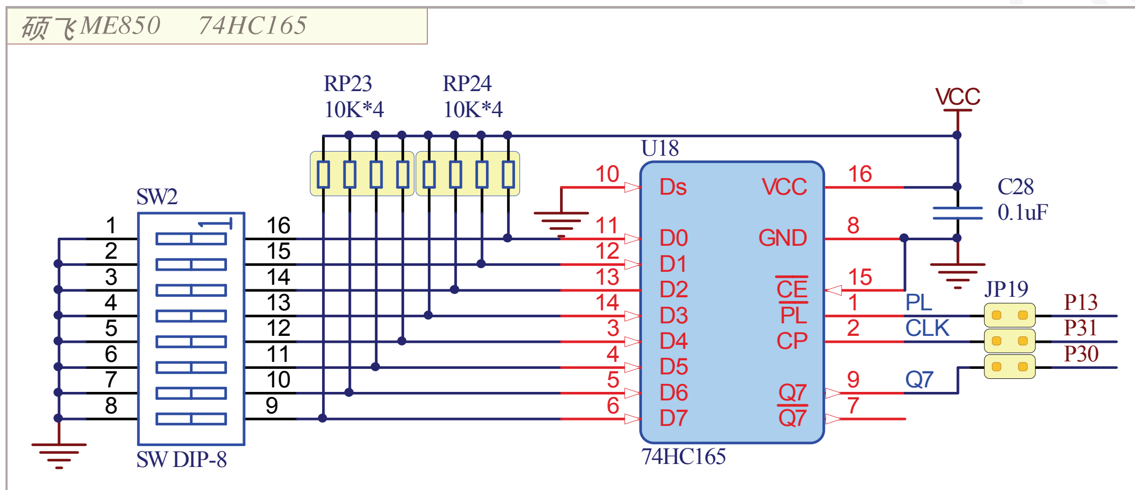


图 5.36 74HC164 原理图

3. 实验步骤

短接 JP19 短接子，使 74HC165 的相应引脚与串行端口（RXD-P3.0、TXD-P3.1）连接。

将 JP9 的 9 个短接子全部用短接帽短接，使 D20～D27 与 P2 端口接通，VCC 向发光二极管模块供电。

将 JP6（步进电机）、JP26（MAX232C）、JP20（74HC164） 短接子上的短接帽全部取掉。

4. 程序流程图

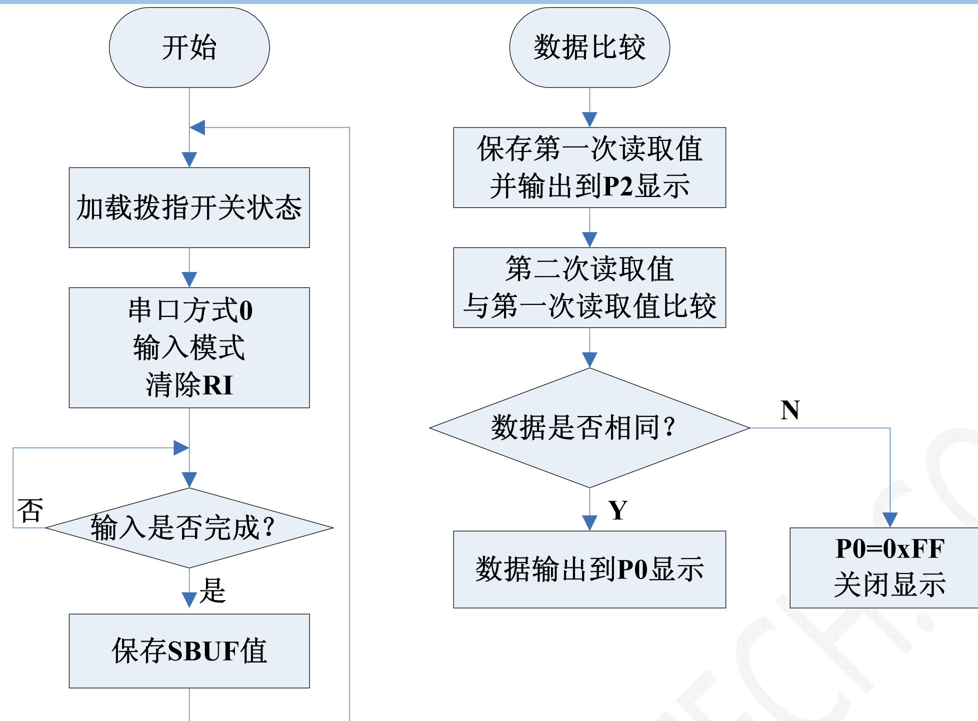


图 5.37　EX14_HC165 流程图

5. 汇编源程序

（见光盘： Example_A51\ EX14_HC165）

6. C 语言源程序

（见光盘： Example_C51\EX14_HC165）

实验十五　步进电机控制

步进电机是一种将电脉冲信号变换成相应的角位移的机电执行元件。控制步进电机的输入脉冲数量、频率及电机各项绕组的接通顺序，可以得到各种需要的运行特性。尤其与数字设备配套时，体现了更大的优越性，因此广泛应用于数字控制系统中。

1. 实验任务

采用单双八拍工作方式，控制步进电机正（顺时针）反（逆时针）方向转动。

步进电机正转 360 度（一圈），反转 180 度（半圈），如此循环。

2. 实验线路

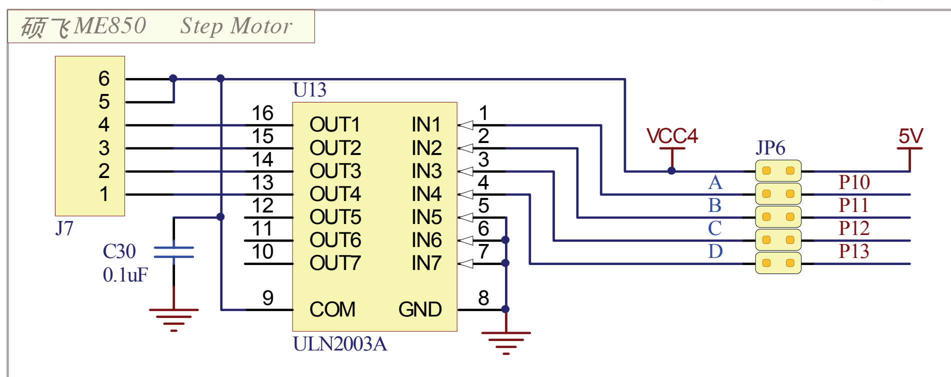


图 5.38　步进电机驱动电路

单片机的 P1.0-P1.3 输出的脉冲信号经 JP6 送到 ULN2003 的 IN1-IN4 输入端，经 ULN2003 放大和倒相后的输出脉冲信号通过 J7 来驱动步进电机作相应的操作。ULN2003 的 COM 端和步进电机的 COM1、COM2 连接到 VCC。步进电机内部原理图如图 5.39 所示。

例如：当单片机的 P1.0 输出高电平时，ULN2003 的 IN1 输入端则为高电平，经过 ULN2003 放大和倒相后在 OUT1 输出端输出低电平，使步进电机的 A 相得电旋转一个步距角。

2.1 ME850 板载步进电机性能指标

2 相 6 线式步进电机

步距角 7.5 度

工作电压>12V（实验时也可以用 5V 供电，只是力矩变小）

额定静力矩＞240g/cm

动力矩＞80g/cm

外形：φ35×15mm

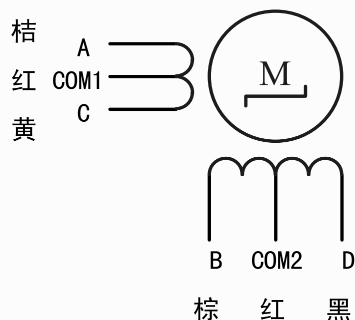


图 5.39　步进电机内部电路

2.2 步进电机内部电路

步进电机电路结构则如图 5.39 所示，包含两组带有中间抽头的线圈，A-COM1-C 为一组，B-COM2-D

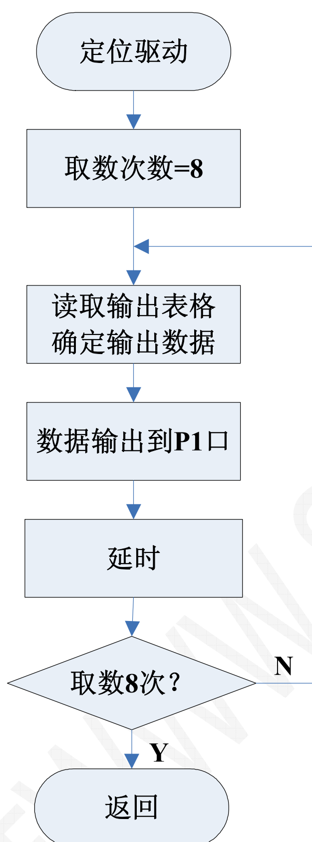
为另一组。整个电机共有六条线。

3. 实验步骤

将 JP6 短接子全部用短接帽短接，使步进电机驱动芯片的输入脚与 P1 端口接通，VCC 向步进电机模块供电。

注意：不做步进电机实验时，请将 JP6 短接子上的短接帽全部取掉，否则将影响其它使用 P1.0-P1.3 端口模块的正常工作。

4. 程序流程图



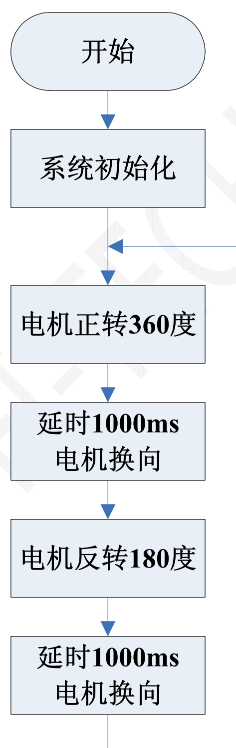


图 5.40　定位子程序流程图　　　图 5.41　主程序流程图

5. 汇编源程序

（见光盘： Example_A51\EX15_Motor）

6. C 语言源程序

（见光盘： Example_C51\EX15_Motor）

实验十六 NE555 计数实验

1. 实验任务

利用 C51 单片机的 T0、T1 定时计数器对 NE555 产生的脉冲信号进行频率计数，频率测量结果通过 LCD1602 显示出来。

信号的频率定义为在 1 秒内信号的周期数，设置 C51 单片机的定时/计数器 0 工作在定时方式，用于实现 1S 定时；设置定时/计数器 1 工作在计数方式，实现对外部时钟脉冲个数的测量。1S 定时结束时，定时/计数器 1 中的计数值即为信号的频率。

2. 实验线路

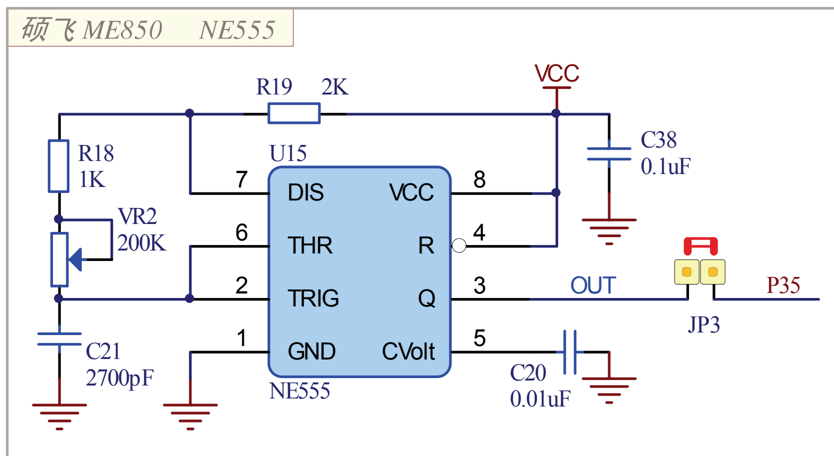


图 5.42 NE555 多谐振荡器电路

3. 实验步骤

将 JP24 的 ON 端短接，使 LCD1602 使能。

将 JP25 背光短接子短接，使 LCD1602 的背光灯点亮。

将 JP3 短接子短接，使 NE555 的输出脉冲信号与 P3.5（T1）连接。

将 JP17（24C04）短接子上的短接帽取掉，否则无法计频。

4. 程序流程图

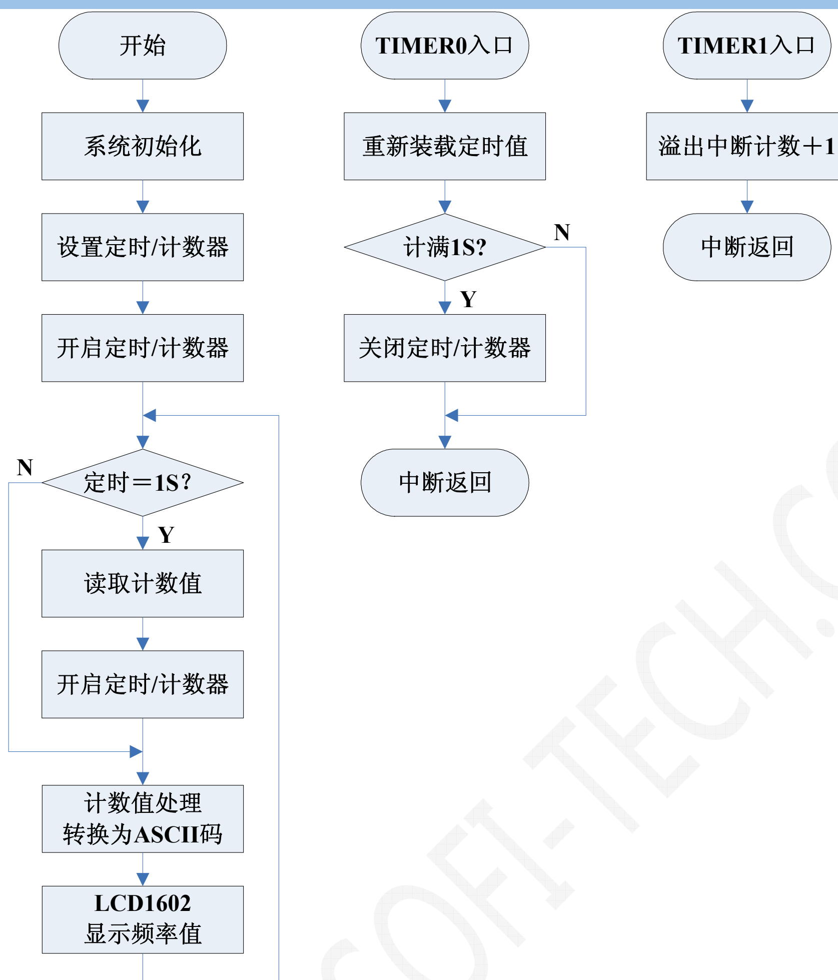


图 5.43　频率计程序流程图

5. 汇编源程序

（见光盘： Example_A51\EX16_NE555）

6. C 语言源程序

（见光盘： Example_C51\EX16_NE555）

实验十七　93C46 读写实验

1. 实验任务

先从地址 0x00 开始连续写入 8 个数据"0-7"，然后再从地址 0x00 开始读出所写入的 8 个数据，存到 8 个显示存储区中，然后再由 8 位数码管显示。写读操作成功后，8 位数码管从右至左依次显示 0-7。

2. 实验线路

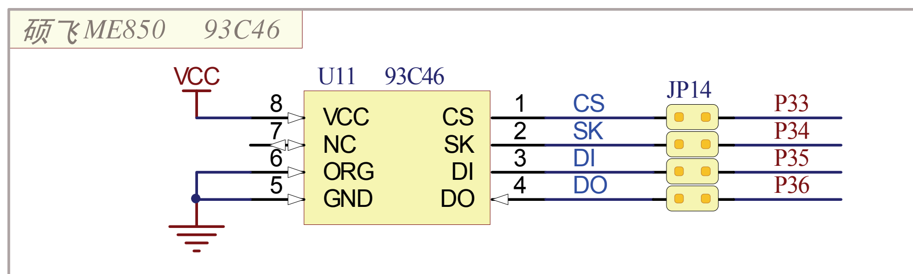


图 5.44　93C46 原理图

3. 实验步骤

将 JP21 的 8 个短接子全部用短接帽短接，使 DG0～DG7 与 P2 端口接通。

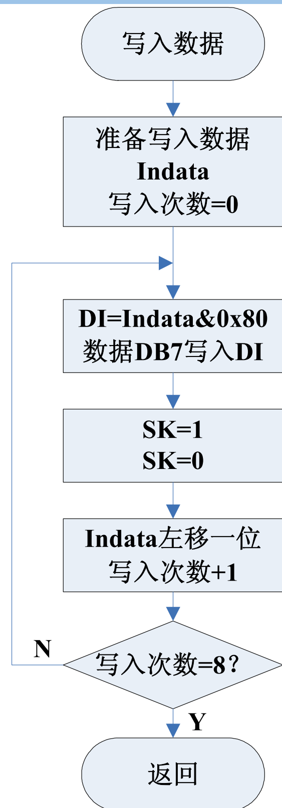
将 JP22 的 9 个短接子全部用短接帽短接，使 A～DP 与 P0 端口接通，VCC 向数码管模块供电。

将 JP14 的 4 个短接子全部用短接帽短接，使芯片管脚与 P3.3～P3.6 端口接通。

将 JP3（NE555）短接子上的短接帽取掉，否则 93C46 无法正常工作。

将 JP24 的 OFF 短接子用短接帽短接，禁止 LCD1602 显示功能，否则数码管将不能正常显示。

4. 程序流程图



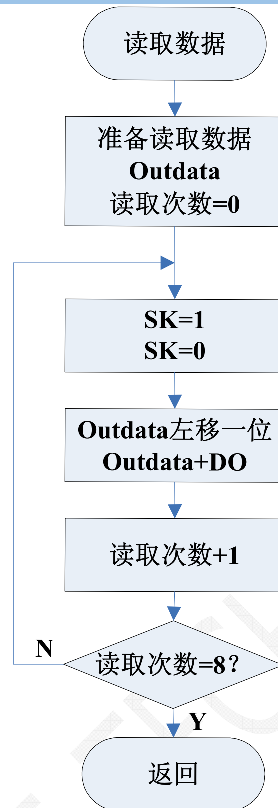


图 5.45　写 1 字节数据流程图　　图 5.46　读 1 字节数据流程图



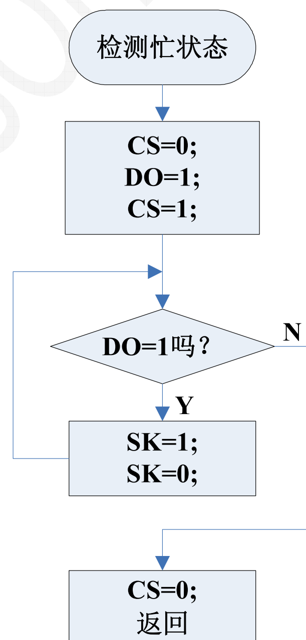


图 5.47　命令与地址写入流程图　　图 5.48　忙检测流程图

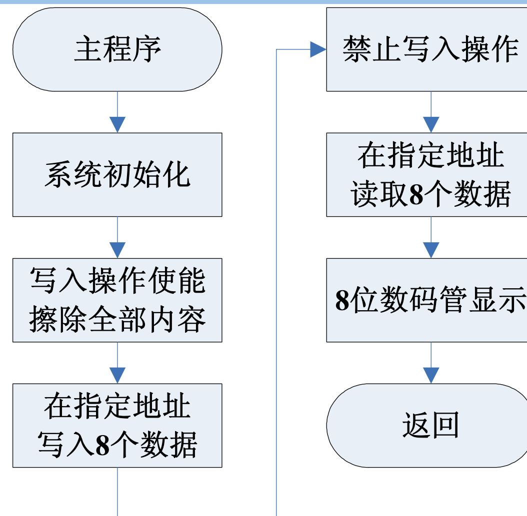


图 5.49　主程序流程图

5. 汇编源程序

（见光盘： Example_A51\EX17_93C46）

6. C 语言源程序

（见光盘： Example_C51\EX17_93C46）

实验十八 24C04 读写实验

1. 实验任务

先从地址 0x00 开始连续写入 8 个数据"0-7"，然后再从地址 0x00 开始读出所写入的 8 个数据，存到 8 个显示存储区中，然后再由 8 位数码管显示。

写读操作成功后，8 位数码管从右至左依次显示 0-7。

2. 实验线路

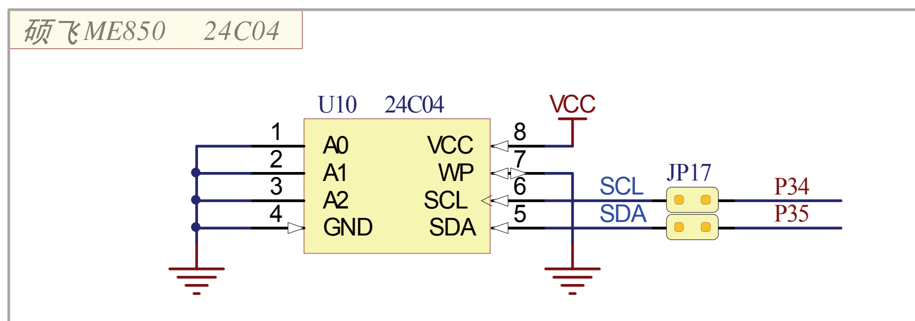


图 5.50 主程序流程图

3. 实验步骤

将 JP21 的 8 个短接子全部用短接帽短接，使 DG0～DG7 与 P2 端口接通。

将 JP22 的 9 个短接子全部用短接帽短接，使 A～DP 与 P0 端口接通，VCC 向数码管模块供电。

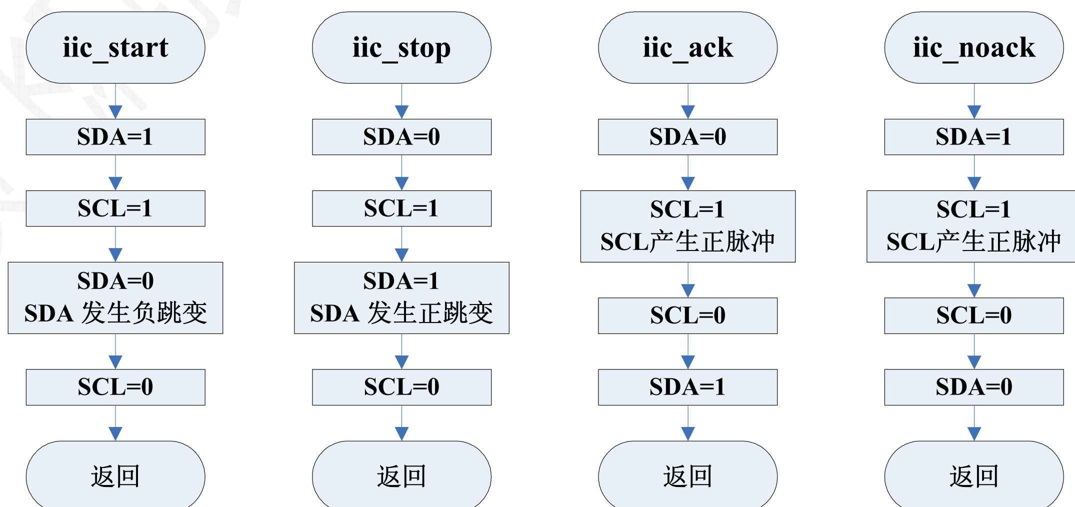
将 JP17 的 2 个短接子全部用短接帽短接，使芯片管脚与 P3.4～P3.5 端口接通。

将 JP3（NE555）短接子上的短接帽取掉，否则 24C04 无法正常工作。

将 JP24 的 OFF 短接子用短接帽短接，禁止 LCD1602 显示功能，否则数码管将不能正常显示。

备注：因为 C51 单片机芯片上没有专用的 I^2C 接口引脚，所以程序只能是采用模拟 I^2C 接口数据传输方式来编写。

4. 程序流程图



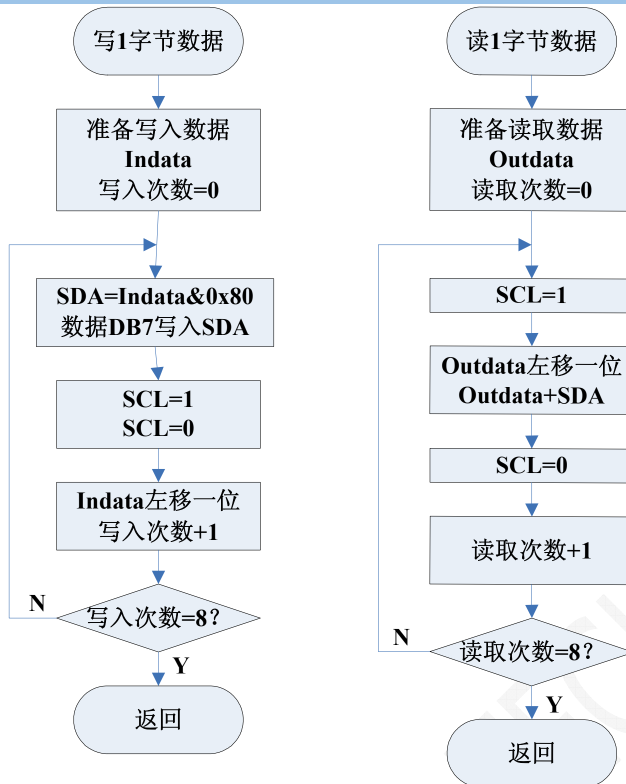


图 5.51 IIC 基础程序流程图

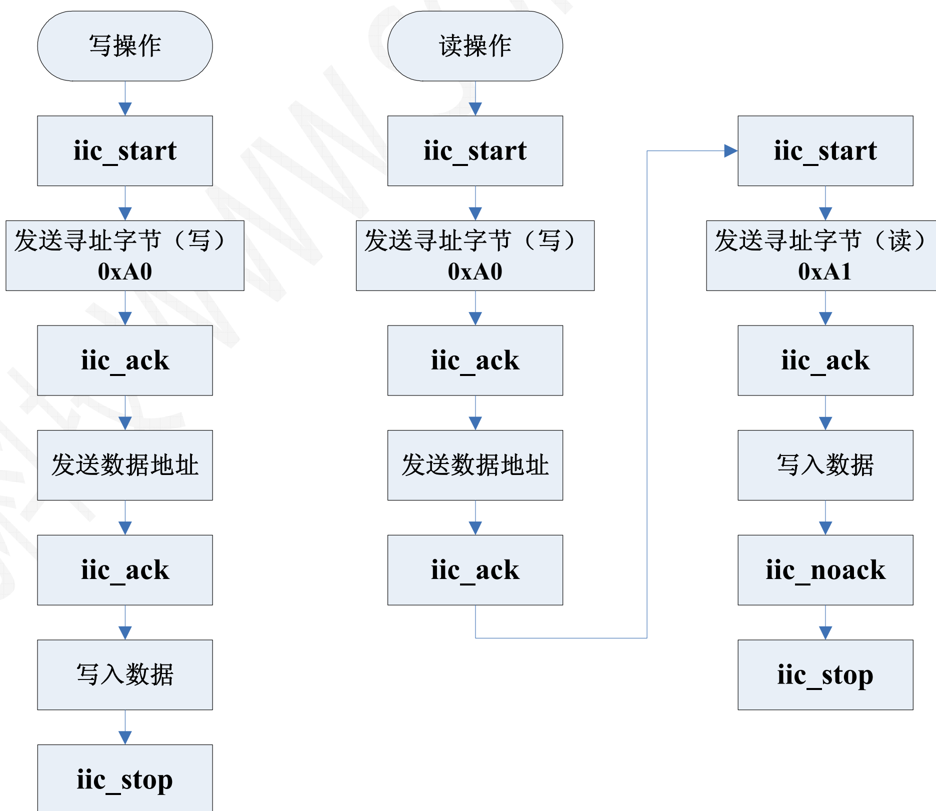


图 5.52 单字节读写操作流程图

5. 汇编源程序

（见光盘： Example_A51\EX18_24C04）

6. C 语言源程序

（见光盘： Example_C51\EX18_24C04）

实验十九　PCF8591 A/D 转换实验

PCF8591 是具有 I2C 总线接口的 8 位 A/D 及 D/A 转换器。有 4 路 A/D 转换输入，1 路 D/A 模拟输出。它既可以作 A/D 转换也可以作 D/A 转换。A/D 转换为逐次比较型。

I2C 总线是 Philips 公司推出的串行总线，整个系统仅靠数据线（SDA）和时钟线（SCL）实现完善的全双工数据传输，即 CPU 与各个外围器件仅靠这两条线实现信息交换。I2C 总线系统与传统的并行总线系统相比具有结构简单、可维护性好、易实现系统扩展、易实现模块化标准化设计、可靠性高等优点。

在一个完整的单片机系统中，A/D 转换芯片往往是必不可少的。PCF8591 是一种具有 I2C 总线接口的 A/D 转换芯片。在与 CPU 的信息传输过程中仅靠时钟线 SCL 和数据线 SDA 就可以实现。

1. 实验任务

4 个 A/D 输入通道：IN0 与电位器 VR1 连接用于测量电位器的输出电压。

IN1、IN2 和 IN3 可测量通过 J13 输入的外接电压。（输入电压要求<=5V）。

D/A 输出：将 IN0 的 A/D 转换值送 D/A 输出，DL10 随 D/A 输出值的大小改变亮度。

LCD1602 同时显示四通道 A/D 转换值，并将 IN0 通道的 A/D 转换值送 D/A 输出。

2. 实验线路

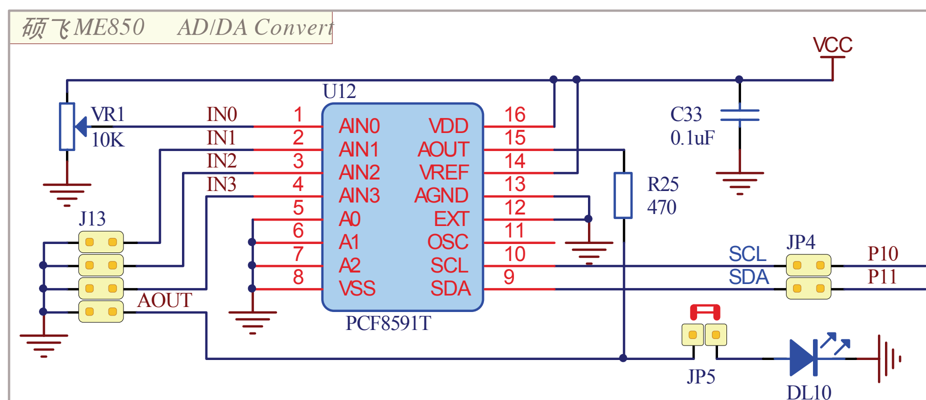


图 5.53　ME850 ADC/DAC(PCF8591)原理图

3. 实验步骤

将 JP24 的 P2.2-EN 短接，使 LCD1602 使能。

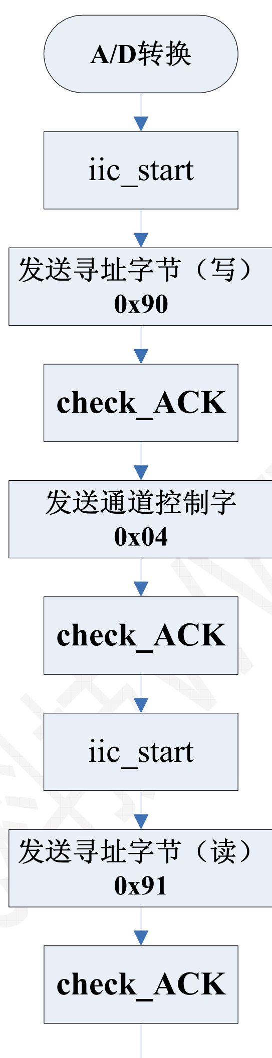
将 JP25 背光短接子短接，使 LCD1602 的背光灯点亮。

将 JP4 的短接子短接，使芯片管脚与 P1.0 ~ P1.1 端口接通。

将 JP5 的短接子短接，使 DL10（LED）与芯片 D/A 输出连接。

注意：如果你不使用 IN1、IN2、IN3 做外接电压输入时，请将 J13 上相应通道的短接子短接，短接后的相应通道输入端与地连接，避免引入干扰。

4. 程序流程图



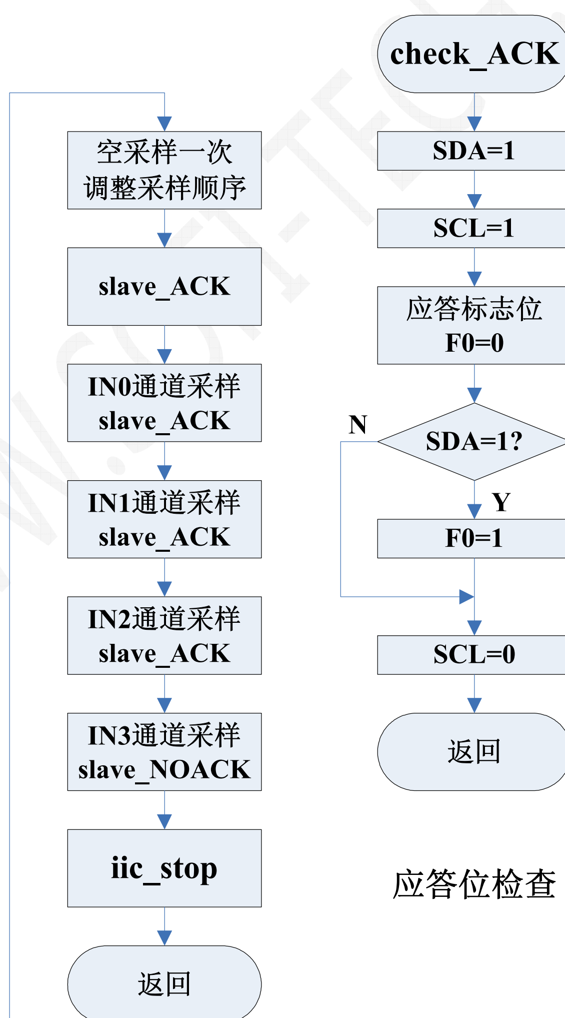


图 5.54　A/D 转换程序流程图

图 5.55　应答位检查程序流程图

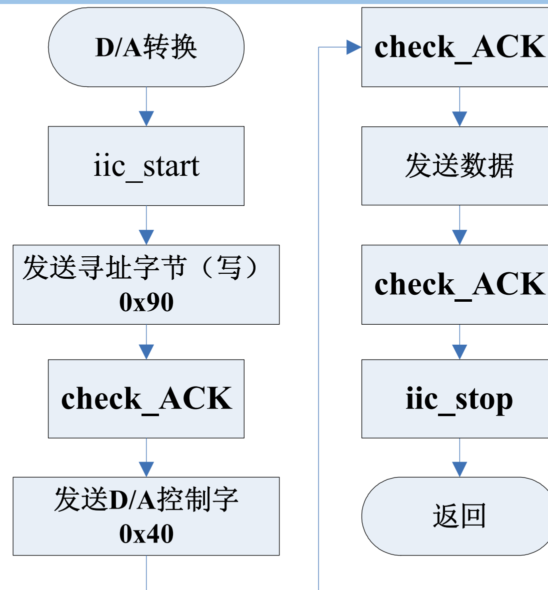


图 5.56　D/A 转换程序流程图

IIC 的基本子程序的流程图参阅 EX17_24C04 程序流程图。

5. 汇编源程序

（见光盘： Example_A51\EX19_ADC）

6. C 语言源程序

（见光盘： Example_C51\EX19_ADC）

实验二十 PCF8591 D/A 转换实验

1. 实验任务

利用 D/A 输出功能，通过输入不同的数据方式，可产生各种波形的输出脉冲。

通过按键 K2 选择输出波形的类形，K3 键启动，K4 键停止。

使用 LCD1602 显示菜单及程序运行过程。

可以使用 DL10（LED）来显示输出脉冲，也可以将示波器探头接在 JP5 的引脚上进行测试和观察。

2. 实验线路

请回顾实验十九的线路

3. 实验步骤

将 JP24 的 P2.2-EN 短接，使 LCD1602 使能。

将 JP25 背光短接子短接，使 LCD1602 的背光灯点亮。

将 JP4 的短接子短接，使芯片管脚与 P1.0～P1.1 端口接通。

将 JP5 的短接子短接，使 DL10 与芯片 D/A 输出连接，用 LED 显示。

如果使用示波器观测时，请将 JP5 短接子上的短接帽取掉，否则波形失真变形。

4. 程序流程图

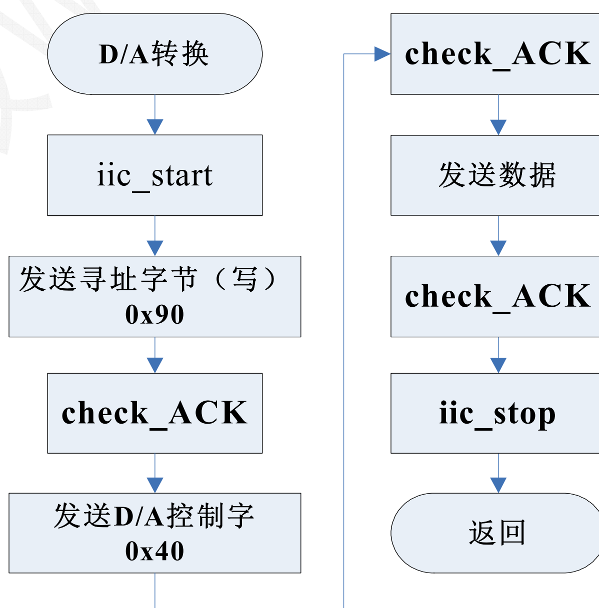


图 5.57 D/A 转换程序流程图

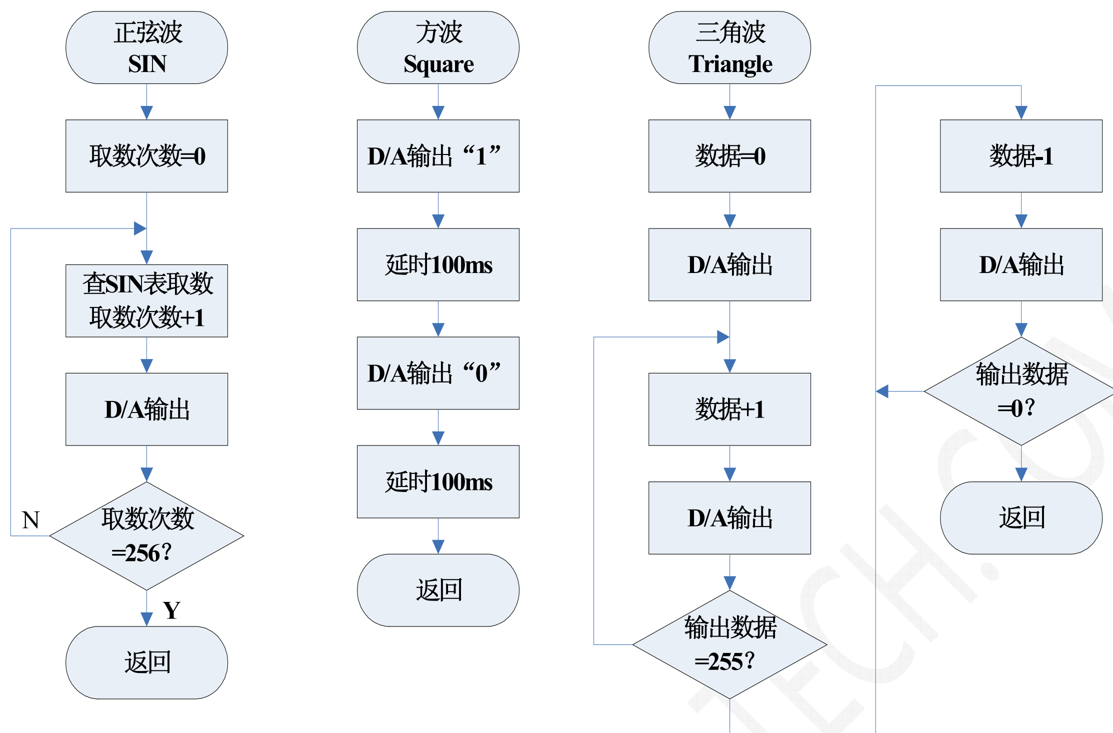


图 5.58　波形输出程序流程图

5. 汇编源程序

（光盘： Example_A51\EX20_DAC）

6. C 语言源程序

（光盘： Example_C51\EX20_DAC）

实验二十一　DS1302 实时时钟

DS1302 是美国 DALLAS 公司推出的一种高性能、低功耗、带 RAM 的实时时钟电路，它可以对年、月、日、周日、时、分、秒进行计时，具有闰年补偿功能，工作电压为 2.5V～5.5V。采用三线接口与 CPU 进行同步通信，并可采用突发方式一次传送多个字节的时钟信号或 RAM 数据。DS1302 内部有一个 31×8 的用于临时性存放数据的 RAM 寄存器。DS1302 增加了主电源/后背电源双电源引脚，同时提供了对后背电源进行涓细电流充电的能力。

1. 实验任务

DS1302 时钟芯片包括实时时钟/日历，提供秒、分、时、日、周、月和年等信息。

用 LCD1602 显示从 DS1302 读出的 年、月、日、星期、时、分、秒 的实时值。

同时按下 K1 和 K4 键将由程序预设的日期和时间数据写入 DS1302 芯片内。

2. 实验线路

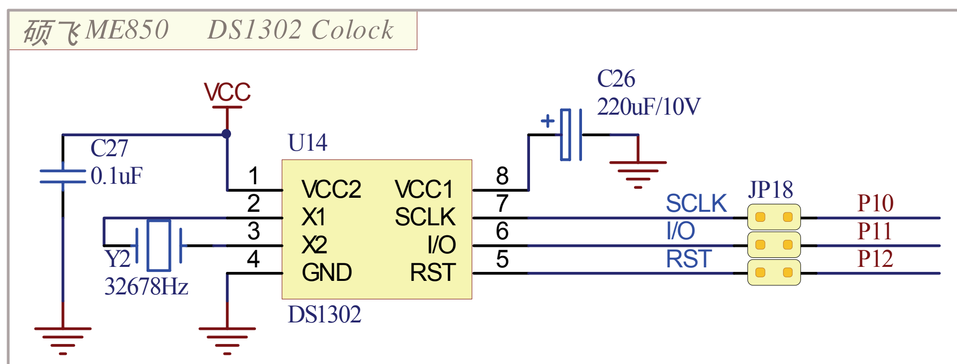


图 5.59　DS1302 原理图

3. 实验步骤

将 JP24 的 ON 端短接，使 LCD1602 使能。

将 JP25 背光短接子短接，使 LCD1602 的背光灯点亮。

将 JP18 短接子短接，使芯片管脚与 P1.0～P1.2 端口接通。

4. 程序流程图

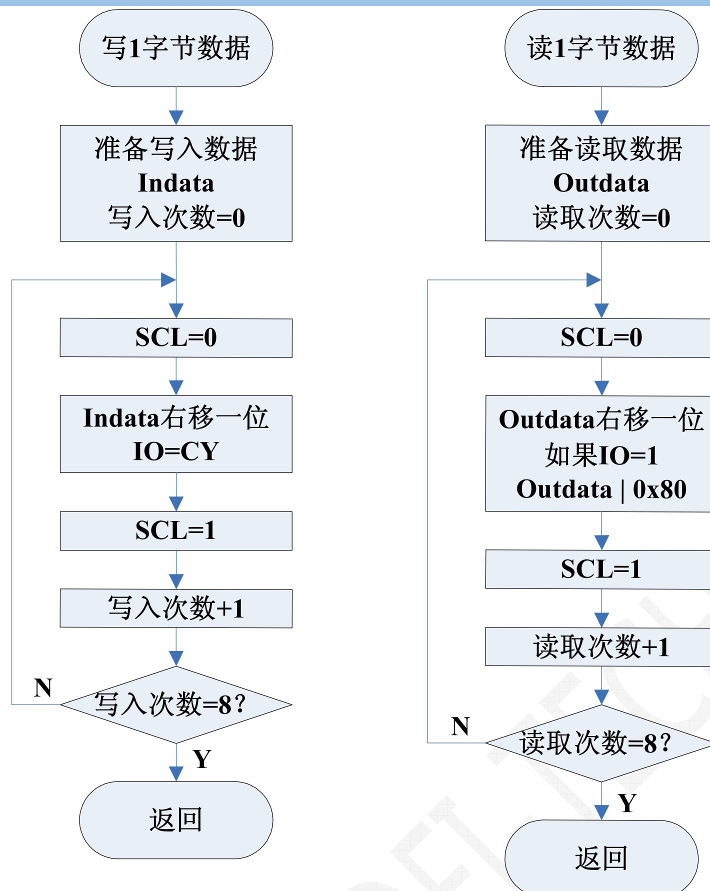


图 5.60　读写 1 字节子程序流程图

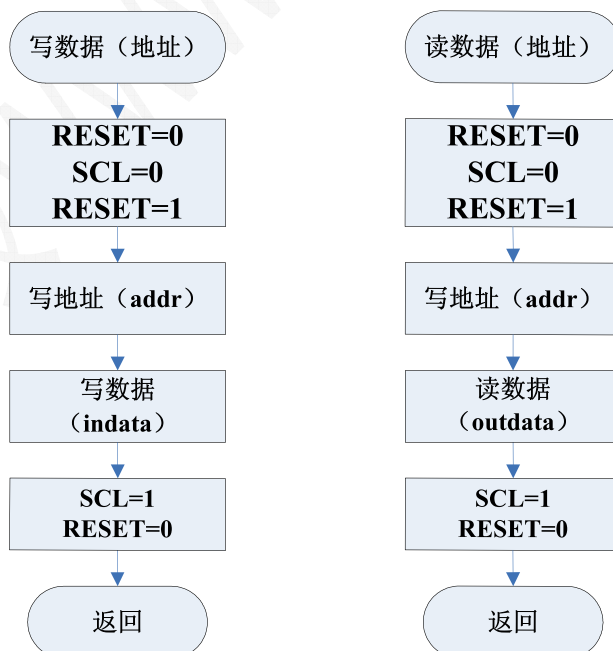


图 5.61　固定地址读写 1 字节程序流程图

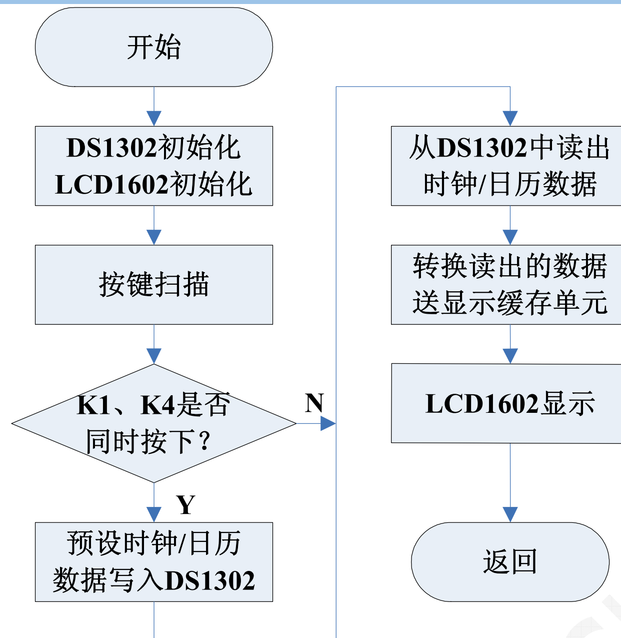


图 5.62　主程序流程图

5. 汇编源程序

（光盘： Example_A51\EX21_DS1302）

6. C 语言源程序

（光盘： Example_C51\EX21_DS1302）

实验二十二　DS18B20 数字温度传感器

DS18B20是Dallas公司生产的单总线(One-wire)数字化温度传感器，它采用单根信号线传输数据，而且数据传输是双向的。它能直接读出被测温度，因此可以通过简单的编程实现温度显示与温度控制。

DS18B20具有如下特性：

- 测温范围： -55 ～ 125 ℃
- 转换精度：9 ～ 12 位（包括符号位），可编程决定转换精度的位数。
- 测温分辨率：9 位精度为 0.5 ℃，12 位精度为 0.0625 ℃。
- 转换时间：9 位精度为 93.75 ms，12 位精度为 750ms。
- 具有非易失性，上、下报警设定功能。

1. 实验任务

DS18B20 检测温度，用 6 位数码管显示实际温度值和符号。

当检测到 DS18B20 不存在或有问题时，蜂鸣器将报警，数码管黑屏。

2. 实验线路

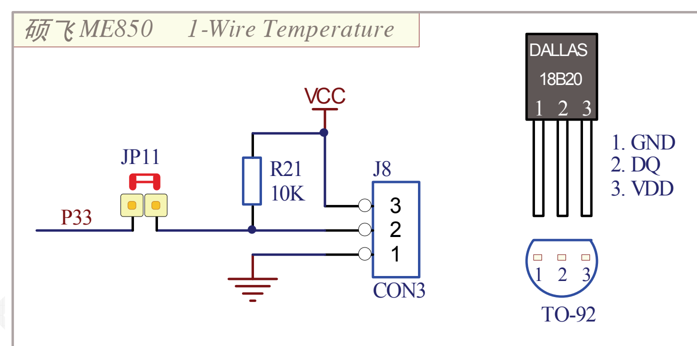


图 5.63　DS18B20 接口原理图

3. 实验步骤

将 JP21 的 8 个短接子全部用短接帽短接，使 DG0～DG7 与 P2 端口接通。

将 JP22 的 9 个短接子全部用短接帽短接，使 A～DP 与 P0 端口接通，VCC 向数码管模块供电。

将 JP11 的短接子用短接帽短接，使 DS18B20 的数据线与 P3.3 端口接通。

将 JP24 的 OFF 短接子用短接帽短接，禁止 LCD1602 显示功能，否则数码管将不能正常显示。

4. 程序流程图

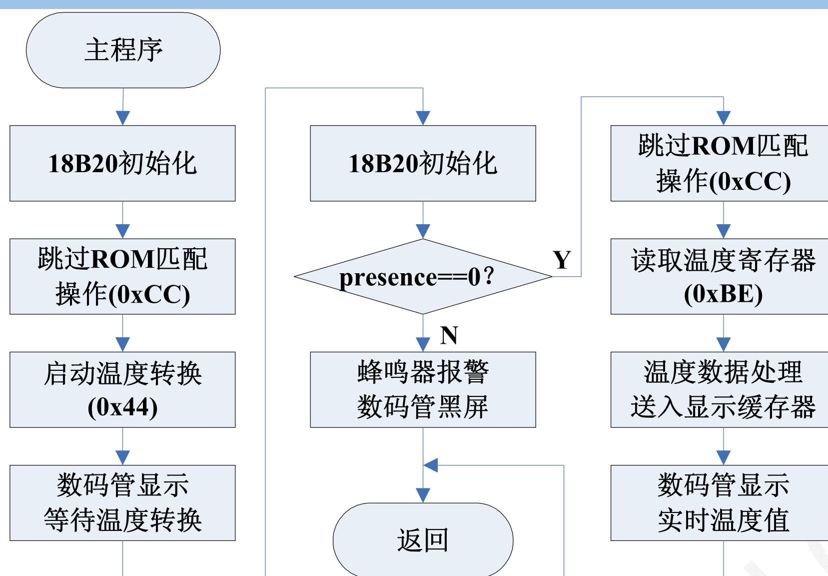


图 5.64　主程序流程图

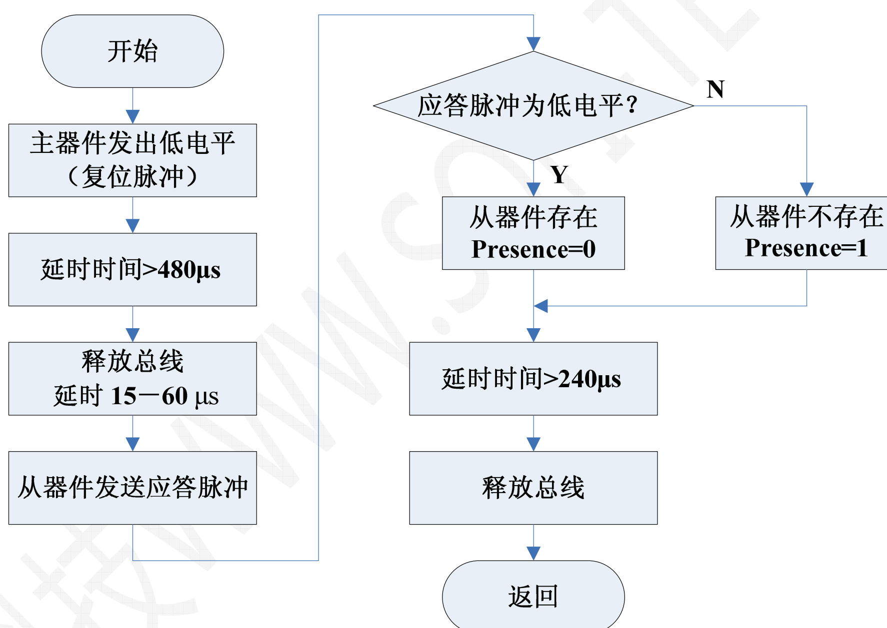


图 5.65　初始化子程序流程图



图 5.66　字节读写子程序流程图

5. 汇编源程序

（光盘： Example_A51\EX22_DS18B20）

6. C 语言源程序

（光盘： Example_C51\EX22_DS18B20）

实验二十三　红外遥控解码实验

现在大部分家电产品都具有遥控功能，如电视机、影碟机、空调、电风扇及音响等。它们都是采用红外线遥控方式来控制，红外遥控器上的每个按键都具有特定的遥控功能。

ME850 单片机开发实验仪集成有一路一体化红外接收头，并配有定制的红外发射器，能够做红外接收与解码实验。

1. 实验任务

红外一体化接收头接收到红外遥控发射器所发射的信号，并将此信号进行整形和反相送入单片机 P3.2 端口。经过软件译码，将译码结果（按键代码）送数码管显示。

2. 实验线路

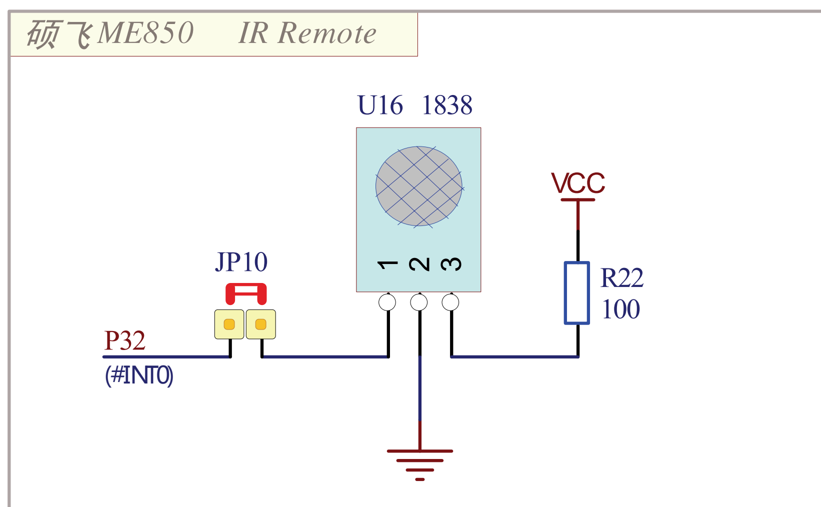


图 5.66　红外接收接口电路

ME850 选用 T1838 一体化红外接收头，接收来自红外遥控器的红外遥控信号。T1838 集成红外接收二极管、放大、解调、整形等电路在同一封装上。T1838 负责红外遥控信号的解调，将调制在 38kHz 上的红外脉冲信号解调并倒相后输入到单片机的 P3.2（INT0）引脚，由单片机进行高电平与低电平宽度的测量（脉冲宽度调制解码）。

T1838 的输出端通过 JP10 与 AT89S52 的 P3.2（INT0）连接，既可以使用中断方式也可以使用查询方式来编程。应用电路如图 5.66 所示。

ME850 配套的红外遥控器采用 DT9122D 芯片制作（兼容 HT6222、SC6122），共有 32 个功能键，在每个按键上标有功能码和此键的数据代码，如图 5.57 所示。当红外遥控器按键按下后，即有规律地将遥控编码发出，所按的键不同，遥控编码也不同。



图 5.57 红外遥控发射器

3. 实验步骤

将 JP21 的 8 个短接子全部用短接帽短接，使 DG0～DG7 与 P2 端口接通。

将 JP22 的 9 个短接子全部用短接帽短接，使 A～DP 与 P0 端口接通，VCC 向数码管模块供电。

将 JP10 的短接子用短接帽短接，使红外接收头 U16 的数据线与 P3.2 端口接通。

将 JP24 的 OFF 短接子用短接帽短接，禁止 LCD1602 显示功能，否则数码管将不能正常显示。

第一次使用遥控器要取下电池盖下的隔离胶片！

4. 程序流程图

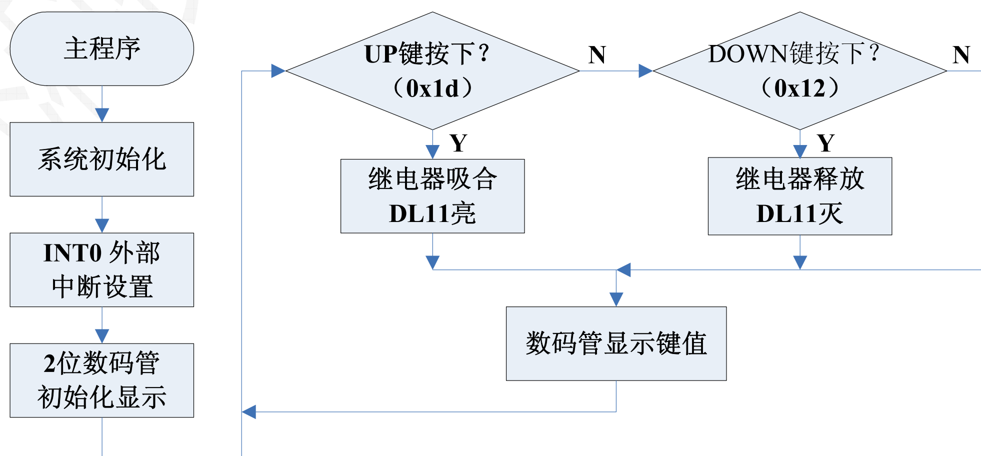


图 5.58 主程序流程图

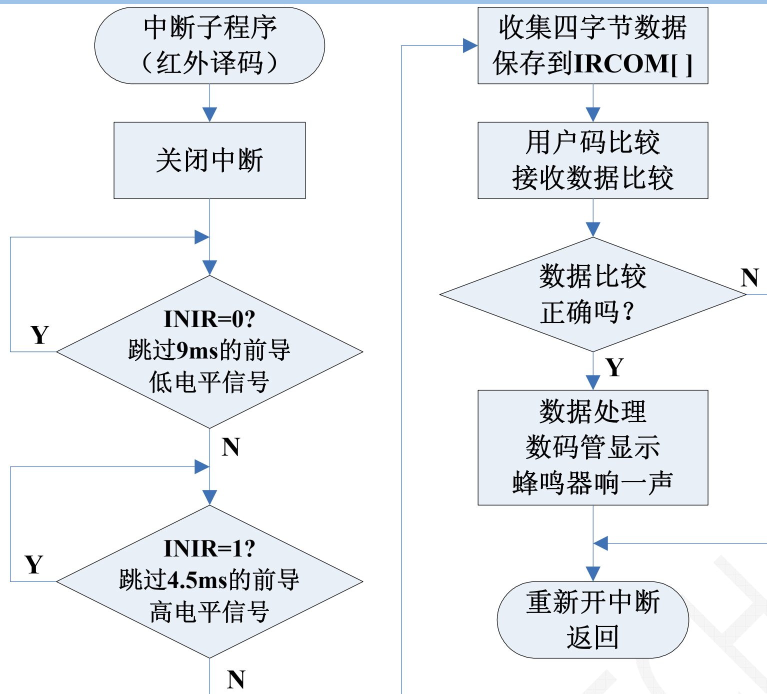


图 5.59　中断子程序流程图

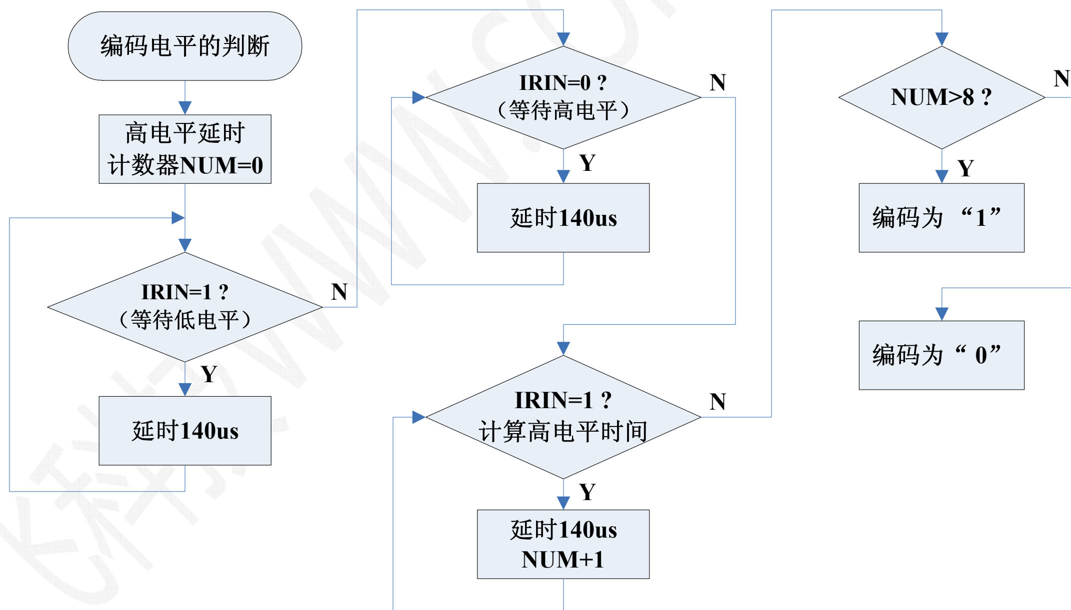


图 5.60　判断编码电平流程图

5. 汇编源程序

（光盘： Example_A51\EX23_IR）

6. C 语言源程序

（光盘： Example_C51\EX23_IR）

实验二十四　PS2 键盘解码实验

1. 实验任务

开机上电时，首先程序向连接在 PS2 接口的 PC 键盘发出复位指令，PC 键盘被复位，你可以看到 PC 键盘上的 3 个 LED 灯闪亮一下，1 位数码管初始显示" - "。

程序定义了 PC 键盘的 1-f 数字键、Esc、NumLock 功能键和右边小键盘中的 0-9 数字键。

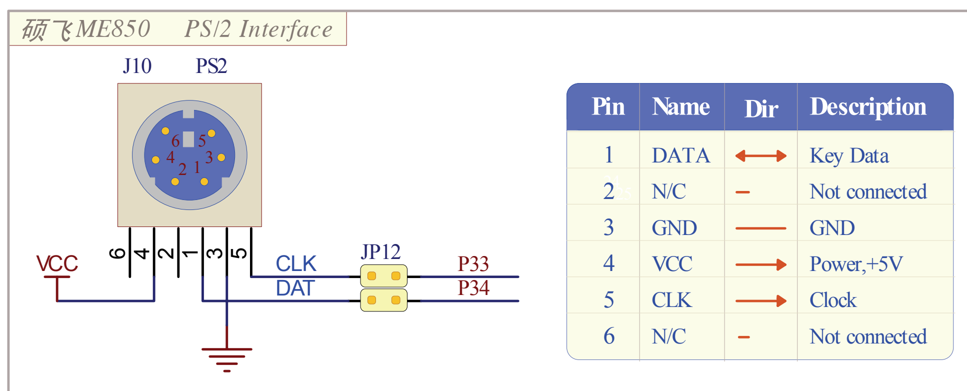
右边小键盘中的 0-9 数字键是否有效由 NumLock 功能键控制，按下 NumLock 功能键，NumLock 灯亮，右边小键盘中的 0-9 数字键有效；按下 NumLock 功能键，NumLock 灯灭，右边小键盘中的 0-9 数字键失效。按下 Esc 键，向 PC 键盘发送复位命令。

按下被定义而且有效的 PC 键盘的按键时，1 位数码管显示其键值，蜂鸣器响一声。

按下 NumLock 功能键，数码管显示"n"，蜂鸣器响一声。

按下没有被定义或被定义但无效的 PC 键盘的按键时数码管显示" - "。

2. 实验线路



Pin	Name	Dir	Description
1	DATA	←→	Key Data
2	N/C	–	Not connected
3	GND	——	GND
4	VCC	→	Power,+5V
5	CLK	→	Clock
6	N/C	–	Not connected

图 5.61　判断编码电平流程图

3. 实验步骤

将 JP21 的 8 个短接子全部用短接帽短接，使 DG0～DG7 与 P2 端口接通。

将 JP22 的 9 个短接子全部用短接帽短接，使 A～DP 与 P0 端口接通，VCC 向数码管模块供电。

将 JP12 的短接子用短接帽短接，使 PS/2 接口线与 P3.3 和 P3.4 端口接通。

将 PC 键盘插入 J10，使 PS/2 接口使能。

将 JP11 的短接子（DS18B20）上的短接帽取掉，以免产生干扰。

将 JP24 的 OFF 短接子用短接帽短接，禁止 LCD1602 显示功能，否则数码管将不能正常显示

4. 程序流程图

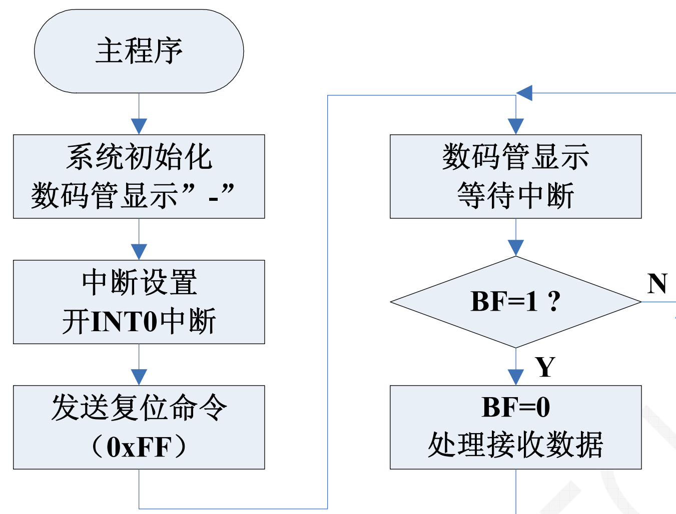


图 5.62　主程序流程图

数据接收编程说明：

PS2 协议采用的传送数据帧由 1 位起始位（0）、8 位数据位、1 位奇偶校验位和 1 位停止位（1）构成。数据发送时低位在前，高位在后。

当 IntNum＝0 时， 起始位（起始位始终为"0"）。

当 IntNum＝9 时， 奇偶校验位

当 IntNum＝10 时，停止位（停止位始终为"1"）

因上述 3 个数据位在数据接收时不考虑进行相应的处理，所以将其丢弃。

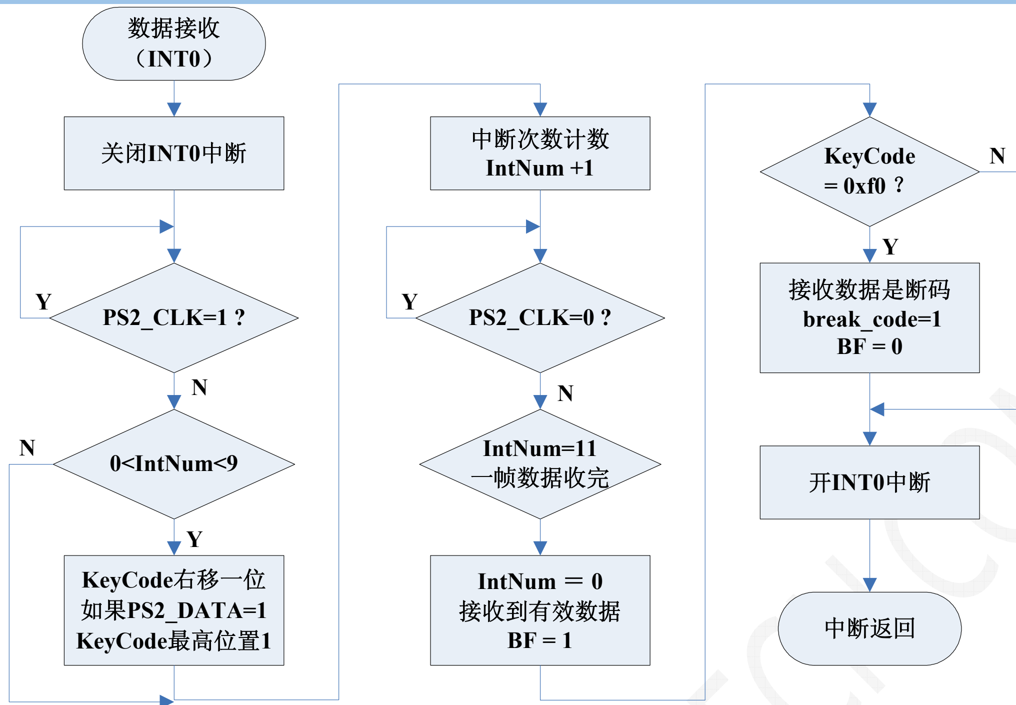


图 5.63　接收数据流程图

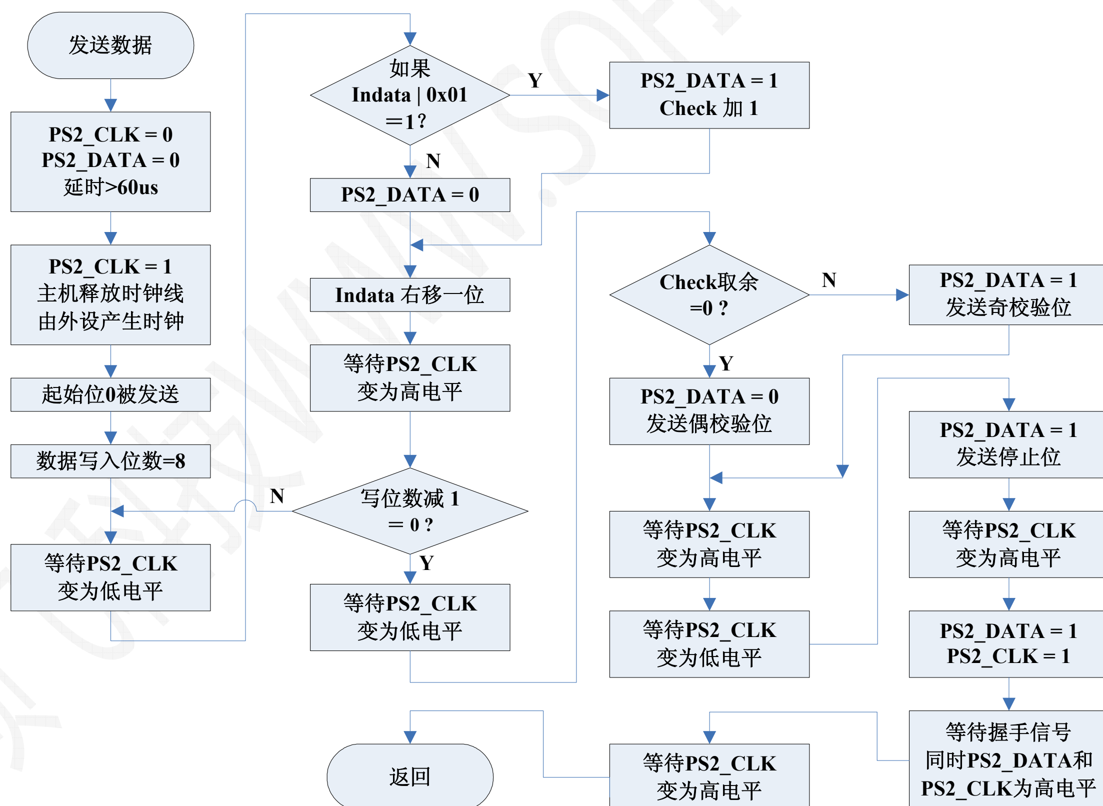


图 5.64　发送数据流程图

5.2 综合实验

本章实验相关文档正在整理中，请留意网站下载更新！

深圳硕飞科技有限公司 Gguoqing

公司网站：WWW.SOFI-TECH.COM

电子邮箱：sofitech@tom.com

2008-10-14

附录 1　常见问题解答

1.为何烧写的芯片不能运行?

烧写的芯片不能运行，通常有以下几个原因：

1)程序问题或者加载的文件不对。如果源代码程序有问题，那么烧写芯片肯定是不能正常工作的。

2)目标板硬件问题：单片机正常工作必须具备相应的硬件条件，例如外部电源，晶振的连接，复位电路等等。

3)编程操作步骤错误（如先读取后编程），正确的操作步骤：擦除->查空->编程->校验。

2.为何有时做数码管、液晶、LED 显示实验的时候不能正常工作？

数码管、1602LCD，12864LCD，16 路 LED 这四个模块默认共用了 P0 口和 P2 口，如果同时使用可能会相互干扰。因此做其中某个模块实验的时候，最好将另外的三个模块断开或禁用。

比如做 1602LCD 或者 12864LCD 显示实验，要将数码管旁边的 JP22 跳线组的 VCC 端短路帽取下，否则液晶可能不能正常工作。同样在做数码管显示实验的时候，为避免 1602LCD 对数码管产生干扰，对于 ME830 直接取下液晶模块即可。ME850 已经固定安装了 1602LCD，只需将 1602LCD 上端的 JP24 跳线短接在 OFF 端就可以了（使用 1602LCD 时须短接在 ON 端）。

如果您需要同时使用 2 组显示模块，可以用杜邦线另外搭配线路。

3.如何设置实验仪上的跳线？

一般按第二章 ME830/850 主板电路布局图的默认位置插上跳线帽即可。使用跳线帽连接硬件的原则是：默认短接的跳线帽可以永久保留不需断开，默认没有插短路帽的实验模块只有在做该模块的实验时才插上，不用时须取下短路帽，避免相互干扰。在 ME830/850 的电路布局图中：表示已经插了跳线帽，表示没有插跳线帽。

特别注意由于 1602LCD 是固定安装在 ME850 主板上的，任何时候不做 1602LCD 实验时一定要将 JP24 的短路帽短接在 OFF 位置来禁用 1602LCD，否则 1602LCD 可能干扰其他使用 P0，P2 口的实验模块；

对于数码管显示模块，不使用时只须断开 JP22 跳线组的 VCC 端跳线；

对于 16x16LED 点阵显示模块、不使用时只须断开 JP23 跳线组的 VCC 端跳线；

对于 LED 显示模块、不使用时只须断开 JP9 跳线组的 VCC 端跳线；

如果您需要自己定义硬件连接，可以取下实验模块的短接跳线帽，然后根据需要用杜邦线连接即可。

4.软件提示器件 ID 错误是什么原因?

每一个器件都有其相应的 ID 信息，软件会通过该特性来检查芯片的型号是否正确。如果芯片的型号选择错误，将会出现此提示。还有一种可能是芯片已经损坏。

5.软件提示器件初始化错误是什么原因?

在对部分器件(例如 AVR 系列单片机)进行烧写时 软件会首先对器件进行初始化操作，如果初始化失败，软件会给出此提示。初始化失败的原因主要有：

1）AVR 芯片的熔丝位配置错误。AVR 系列芯片有一些熔丝位，如果设置不当将导致该芯片不能采用 ME800 系列开发实验仪进行 ISP 编程操作。这种情况下只能通过其他并行编程器将该熔丝位进行恢复，导致不能进行编程操作的熔丝位包括：

- ISP 编程使能熔丝位
- RESET 端口被设置为 IO 模式
- 时钟熔丝位被配置位不匹配的模式，例如选择了外部时钟源或者外部 RC 震荡，而单片机外部又没有相应的电路或电子元件.

2）ISP 连接错误. 在进行外部 ISP 编程操作，必须保证 ISP 的连线正确；

3）芯片有问题；

4） 芯片没有内置相应的 ISP 启动代码，在对 SST/WINBOND/NXP 等公司的器件进行编程时，芯片内部必须首先内置有相应的 ISP 启动代码（光盘中已经附有相关启动代码），否则芯片将不能进行编程操作。

6.什么是 ISP ?

ISP 是 In System Programming 的简称，即在系统中编程。这种编程方式无需将芯片从线路板上取下来，仅仅需要连接几条的烧写控制线即可，用户在设计目标板时，预留相应的 ISP 接口，即可随时通过 ISP 更新单片机内的程序。

ME800 系列实验仪均支持 ISP 编程操作，使用方法见第四章的介绍。注意：在进行 ISP 下载时，选择的器件型号必须带有@ISP 后缀。

7.为什么 C 语言中有些代码行不能设置断点?

C 语言具有优化功能，如果当前的代码被优化掉了，则在此行代码前是不能设置断点的。

8.如何得到技术支持以及可以得到哪些技术支持？

一些初次使用中的问题可以在技术支持网站(www.mcusj.com 单片机世界 或者从 www.willar.com 伟纳电子网登录）论坛相关版块搜索，常见问题大多有解答。如果没有找到答案，建议您将所遇到的问题在论坛发帖提出，发帖时详细描述所遇到的问题，必要时配以图示，我们有安排工程师负责解答论坛技术问题，也有很多的热心网友和您一起学习交流。您也可以通过 Email、电话联系。由于电话中不容易说明白问题，非紧急情况下建议通过电话之外的其他途径联系。

我们可提供与 ME800 产品、实验例程相关的技术咨询，但不负责用户项目和实验的技术支持。